Filipe Daniel Sousa dos Anjos

Licenciado em Ciências da Engenharia Electrotécnica
e de Computadores

eVentos 5 - Introdução de melhorias em controlador para navegação autónoma de pequeno veleiro

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador: Doutor Luís Filipe dos Santos Gomes

Professor Associado com Agregação, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri:

Presidente: Doutora Ana Inês da Silva Oliveira

Professora Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Arguente: Doutora Anikó Katalin Horváth da Costa

Professora Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Vogais: Doutor Luís Filipe dos Santos Gomes

Professor Associado com Agregação, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Novembro, 2020

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

eVentos 5 - Introdução de melhorias em controlador para navegação autónoma de pequeno veleiro

A todos aqueles que face a um período de crise, pandemia e desemprego se recusam a dar-se como vencidos.

Agradecimentos

Deve-se o devido agradecimento ao professor Luís Filipe dos Santos Gomes pelo tempo disponibilizado e conhecimentos prestados ao longo da elaboração desta dissertação.

Também um especial agradecimento ao meu pai e todos aqueles que me apoiaram na tomada de decisão de realizar o reingresso académico com vista à conclusão do mestrado em Engenharia Electrotécnica e de Computadores.

Resumo

A fim de se construir um sistema de controlo capaz de permitir a navegação autónoma de uma embarcação à vela em miniatura, a Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa deu início a uma linha de dissertações denominada eVentos.

Sendo que o trabalho até agora concretizado se encontra distribuídos ao longo de vários projetos de autorias distintas, existe a necessidade de uma estratégia de controlo que combine as soluções existentes de forma a se obter um sistema funcional.

Esta dissertação trata da análise, melhoria e junção das soluções presentes nas dissertações eVentos 2, eVentos 3 e eVentos 4 com vista à criação de um sistema de controlo unificado capaz de permitir a navegação autónoma de um veleiro em miniatura ao longo de várias etapas pré-definidas, quer em condições de vento favorável ou desfavorável, assim como a criação de sistemas periféricos para fins de controlo remoto, feedback em tempo real e registo de resultados.

Palavras-chave: Arduino, bússola magnética, cata-vento, controlo difuso, GPS, navegação à bolina, navegação autónoma, rede de Petri, servo, transmissão por radiofrequência, veleiro.

Abstract

In order to build a control system capable of allowing a miniature sailboat to navigate in an autonomous fashion, the NOVA School of Science and Technology began a series of dissertations denominated eVentos.

Given that the work accomplished so far is distributed among several projects of distinct authorships there is a need for a control strategy that combines all the existent solutions in order to obtain a fully functional system.

This dissertation attends to the analysis, improvement and merging of the solutions present in three previous dissertations named eVentos 2, eVentos 3 and eVentos 4 with the goal of creating an unified control system capable of allowing a miniature sailboat model to navigate autonomously along several pre-defined waypoints, both in favorable and unfavorable wind conditions, along with the creation of peripheral systems to ensure remote control capability, real time feedback of the controller's actions and data logging.

Keywords: Arduino, autonomous navigation, fuzzy control,. GPS, magnetic compass, Petri net, radio frequency transmission, sailboat, servo, tacking, windvane.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas

- **AC/DC: A**lternating **C**urrent/**D**irect **C**urrent
- **CINAV: C**entro de **I**nvestigação **Nav**al da Marinha Portuguesa
- **COA: C**enter **o**f **A**rea
- **DC: D**irect **C**urrent
- **EDT: E**vent **D**ispatch **T**hread
- **EEPROM: E**lectrically-**E**rasable **P**rogrammable **R**ead-**O**nly **M**emory
- **FCT/UNL: F**aculdade de **C**iências e **T**ecnologia da **U**niversidade **N**ova de **L**isboa
- **g: g**rama
- **GND: G**rou**nd**
- **GPS: G**lobal **P**ositioning **S**ystem
- **GRES:** R&D **G**roup on **R**econfigurable and **E**mbedded **S**ystems
- **GUI: G**raphical **U**ser **I**nterface
- **I/O: I**nput/**O**utput
- **I2C: I**nter **I**ntegrated **C**ircuit
- **IDE: I**ntegrated **D**evelopment **E**nvironment
- **KB: K**ilo**b**yte

- **Kg: K**ilo**g**rama
- **KML: K**eyhole **M**arkup **L**anguage
- **m: m**etro
- **mA: m**ili**a**mpere
- **MinGW: Min**imalist **G**NU for **W**indows
- **mm: m**ilí**m**etro
- **MOM: M**ean **o**f **M**aximum
- **ms: m**ili**s**segundo
- **NMEA: N**ational **M**arine **E**lectronics **A**ssociation
- **PC: P**ersonal **C**omputer
- **PWM: P**ulse **W**idth **M**odulation
- **RdP: R**ede **d**e **P**etri
- **REX: R**obotics **Ex**ercise
- **RF: R**adio**f**requência
- **SRAM: S**tatic **R**andom **A**ccess **M**emory
- **TTL: T**ransistor-**T**ransistor **L**ogic
- **UDP: U**ser **D**atagram **P**acket
- **USB: U**niversal **S**erial **B**us
- **V: V**olt
- **WGS: W**orld **G**eodetic **S**ystem
- **Wi-Fi: Wi**reless **Fi**delity
- **WRSC/IRSC: W**orld **R**obotic **S**ailing **C**hampionship and **I**nternational **R**obotic **S**ailing **C**onference

Capítulo 1

Introdução

1.1. Enquadramento e motivação

A navegação autónoma é uma área complexa e extensa que engloba várias técnicas de aquisição sensorial, algoritmia e comunicação à distância. Apesar disto ela revela um enorme potencial em termos de aplicações possíveis tais como operações de salvamento, monitorização do ambiente marítimo e outras atividades de natureza oceanográfica [1].

Dos tipos de embarcação existentes, a embarcação à vela apresenta várias vantagens em comparação com as embarcações motorizadas, como por exemplo a poupança energética inerente da utilização do vento e o baixo preço comparativamente a outras técnicas de locomoção utilizadas.

Foi com o interesse na área de navegação autónoma de pequenos veleiros que a Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa deu início à série de projetos denominados eVentos, da qual a presente dissertação dá continuidade, a fim de explorar arquiteturas, metodologias e utilidades para este tipo de sistemas.

1.2. Problema

A linha de projetos eVentos tem como objetivo final a realização de um sistema de controlo capaz de permitir que um modelo de pequeno porte de uma embarcação à vela seja capaz de navegar ao longo de um trajeto composto por várias etapas pré-definidas, quer esta se encontre em condições de vento favorável ou desfavorável, isto é, que seja capaz de completar uma regata.

A fim de se conseguir este objetivo realizaram-se três dissertações distintas que definiram princípios teóricos, estratégias e arquiteturas de hardware a fim de atingir a finalidade proposta:

- eVentos 2 - Autonomous sailboat control [2];
- eVentos 3 - Desenvolvimento de controlador difuso para navegação autónoma de veleiro [3];
- eVentos 4 - Controlador para navegação autónoma de veleiro em modo de regata [4].

Dado que isto é um trabalho contínuo desenvolvido por vários autores, há a falta de uma visão global consolidada, quer do ponto de vista de código desenvolvido por cada dissertação individual, quer em termos de documentação, o que torna por vezes a interpretação e utilização dos meios desenvolvidos uma tarefa complicada.

1.3. Objetivos

Esta dissertação apresenta como objetivo a reformulação, melhoria e modularização do sistema de controlo desenvolvido pela linha de projetos eVentos a fim de se obter um sistema de controlo unificado que seja fiável, funcional e com documentação pormenorizada que facilite a continuação de projetos futuros que utilizem quer o sistema completo desenvolvido ou apenas porções deste.

Esta dissertação conta com os seguintes desafios:

- **Revisão e melhoria de estratégias desenvolvidas:** a fim de se obter um sistema de controlo fiável e funcional há que rever, melhorar e organizar as estratégias propostas;
- **Modularização de código:** o código desenvolvido deve ser organizado por módulos a fim de permitir a sua reutilização em projetos futuros relacionados com a linha eVentos;
- **Produção de documentação pormenorizada:** devido à escassez de informação de projetos anteriores relacionada com esquemas de ligações, passos de calibração e características do código de controlo, deve-se produzir documentação criteriosa referente a todos os componentes do sistema, de modo a conseguir-se uma rápida compreensão do trabalho realizado para reutilização em projetos futuros;
- **Adição de novos componentes:** além dos pontos anteriores, esta dissertação adicionará dois componentes periféricos, um com a função de permitir o controlo manual remoto por parte do utilizador e outro para permitir a leitura do estado do controlador em tempo real, assim como o registo de valores do mesmo.

1.4. Estrutura do documento

Este documento é composto por um total de 7 capítulos de conteúdo mais um capítulo referente à bibliografia.

O primeiro e presente capítulo faz uma introdução em que se descreve a origem da linha de projetos eVentos assim como os objetivos a que a presente dissertação se propõe.

O segundo capítulo faz uma apresentação de alguns conceitos teóricos necessários à compreensão do trabalho desenvolvido ao longo desta dissertação, descrevendo o modelo de embarcação utilizada e alguma terminologia náutica.

O terceiro capítulo dedica-se a uma análise do estado da arte, explicando as características dos sistemas de controlo definidos nas dissertações passadas da linha eVentos, servindo estas de referência à realização desta dissertação.

O quarto capítulo descreve as características dos componentes de hardware e software utilizados ao longo da elaboração desta dissertação, tais como a plataforma de controlo utilizada, sensores, atuadores e plataformas de geração de código de controlo.

O quinto capítulo faz a descrição pormenorizada do sistema de controlo desenvolvido, explicando a sua arquitetura, estratégia de controlo, assim como uma descrição de cada componente individual do sistema.

O sexto capítulo descreve os testes realizados com o sistema de controlo para fins de validação do seu funcionamento, descrevendo os métodos e trajetos selecionados para o efeito assim como a listagem e análise dos resultados obtidos.

O sétimo capítulo trata da conclusão, fazendo uma listagem de propostas referentes a projetos futuros que possam vir a melhorar o desempenho do controlador assim como possíveis soluções a problemas que possam ocorrer em contextos de navegação.

Capítulo 2

Modelo utilizado e conceitos teóricos

Este capítulo faz o levantamento de alguns conceitos teóricos necessários à compreensão do trabalho desenvolvido nesta dissertação, incluindo alguma terminologia marítima, princípios relacionados com a lógica difusa, redes de Petri e a descrição dos modelos de embarcação utilizados na linha eVentos.

2.1. Frota eVentos: modelo utilizado

A frota laboratorial eVentos é composta por três modelos em miniatura de um iate recreativo Naulantia 1M Racing Yacht [5] de aproximadamente um metro de comprimento, do fabricante Tailandês de modelos telecomandados Thunder Tiger, presentes na figura 2.1.

Cada modelo é distinguível por uma cor diferente, contendo também diferentes caixas de ligações onde convergem os fios dos diferentes sensores a fim de ligar à plataforma que contém o controlador.

Figura 2.1: Frota eVentos [5].

2.2. Terminologia marítima

O conhecimento das partes constituintes da embarcação e da terminologia associada com manobras desempenhadas em função do vento são essenciais para o desenvolvimento de estratégias autónomas de navegação.

2.2.1. Constituintes da embarcação

A embarcação é constituída por um mastro que se eleva do casco, tendo este duas velas adjacentes: a vela frontal denominada estai que se estende desde o mastro até à proa e a vela principal que se estende até à popa. Estas duas são controladas conjuntamente por ação de um servo conectado com cordas a ambas as velas. Além disto tem também um leme responsável pela mudança de direção no plano horizontal ("bearing" ou "yaw"), sendo este também controlado por um servo.

Em baixo do casco existe uma plataforma vertical que se estende desde o mesmo até abaixo da linha da água denominada patilhão, responsável tanto por impedir uma rotação em redor do eixo horizontal que passa ao longo do casco no sentido proa-popa, chamado "roll", como também de ajudar a evitar possíveis desvios de rota causados pela força do vento.

O movimento de rotação ocorrido em torno do eixo horizontal no sentido bombordo-estibordo designa-se de "pitch", contudo este não será relevante para fins de controlo. A indicação destes termos encontra-se na figura 2.2.

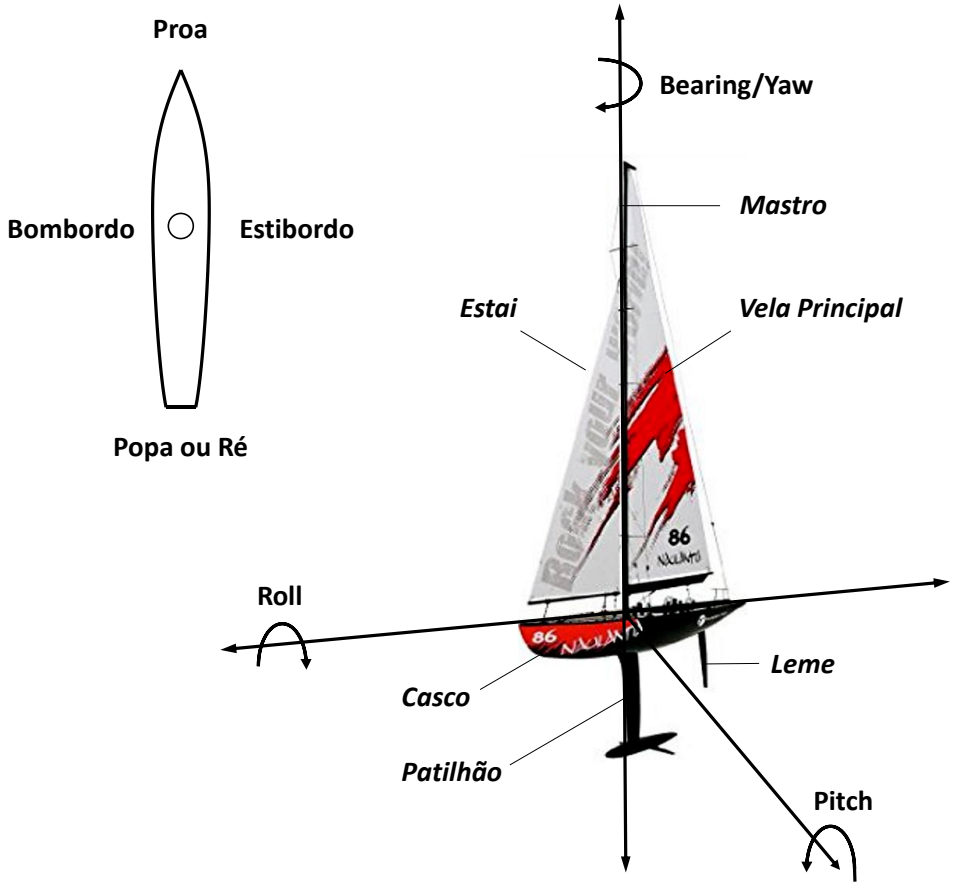


Figura 2.2: Denominação de bordos *à esquerda*; Constituintes da embarcação e movimentos de rotação *à direita* [5], [6].

2.2.2. Vento

Sendo que o modelo utilizado é uma embarcação à vela, convém ter conhecimento prévio das características de navegação associadas com o vento.

2.2.2.1. Vela náutica

A vela náutica é um meio de propulsão marítima capaz de gerar a locomoção de uma embarcação através da criação de uma zona de alta pressão na área direcionada ao sentido do vento (lado barlavento), ao mesmo tempo que gera uma zona de baixa pressão na área oposta (lado sotavento) [6], [7].

Além de propulsionar a embarcação, o vento também pode ser responsável por uma força lateral aerodinâmica que resulta numa deriva lateral da embarcação, apesar do patilhão contribuir para a sua redução [7].

2.2.2.2. Vento aparente

No decorrer da navegação há que ter em conta o facto de que o vento sentido na embarcação, denominado de vento aparente (V_A), é a soma vetorial entre o vento real (V_R) resultante das condições climatéricas e o vento induzido (V_I) causado pelo movimento da embarcação, de sentido oposto ao rumo [7], [8], conforme descrito na fórmula 2.1 e figura 2.3.

$$\mathbf{V_A} = \mathbf{V_R} + \mathbf{V_I} \tag{2.1}$$

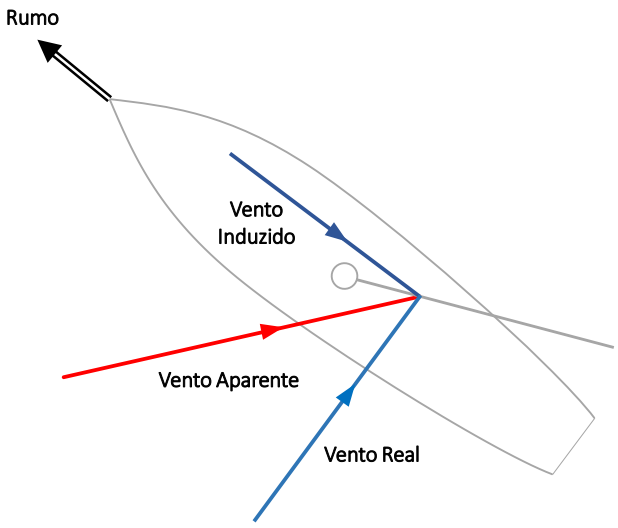


Figura 2.3: Vento aparente e seus constituintes vetoriais [7], [8].

2.2.2.3. Mareações

A maneira consoante um veleiro navega em função ao vento pode ter variadas designações. A estes estados, representados na figura 2.4, dá-se o nome de mareações [7].

O ato de virar uma embarcação no sentido do vento tem o nome de orçar e ao ato de virar a embarcação para longe do mesmo chama-se arribar. Quando o vento incide no sentido dos bordos do veleiro diz-se que este navega de través e, caso este incide na popa, a mareação terá esse mesmo nome.

Se o veleiro navegar a um pequeno ângulo além do estado de vento de popa diz-se que este navega ao largo e se navegar num pequeno ângulo além de través, no sentido do vento, diz-se que este navega à bolina. Além da zona de bolina, ao se aproximar do sentido do vento, situa-se a zona proibida.

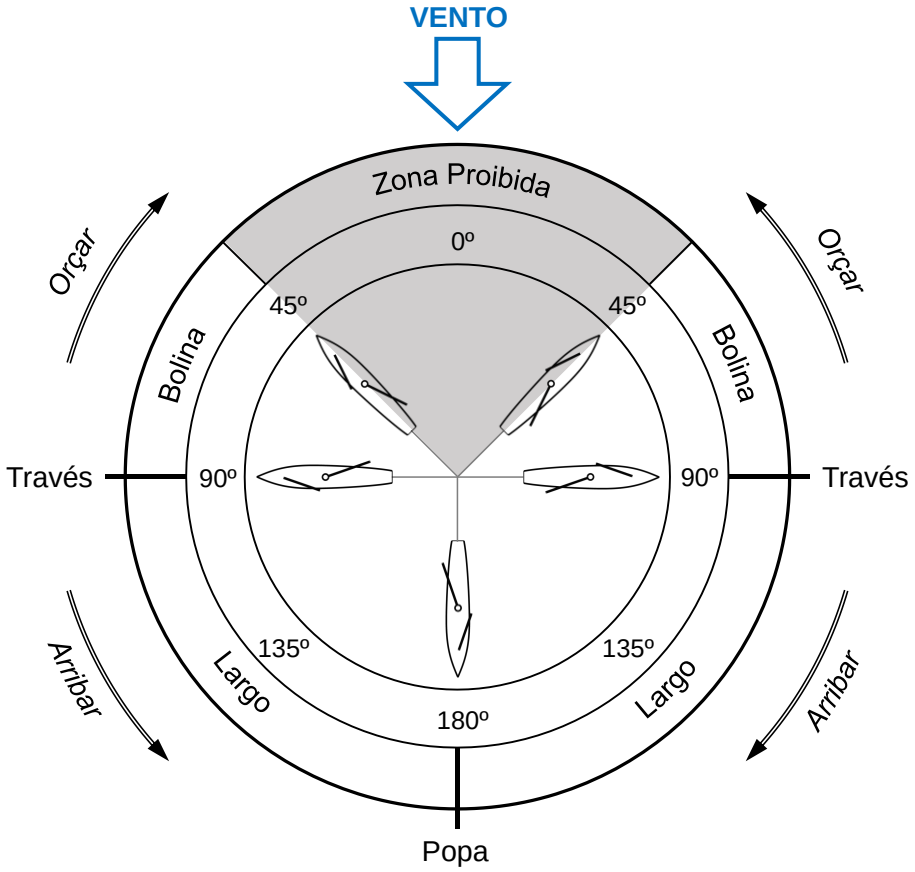


Figura 2.4: Mareações [7].

Uma vez que é impossível navegar contra o vento, define-se uma zona proibida de navegação dentro de um intervalo angular simétrico centrado na direção do vento, considerando-se como referência a zona abrangida desde 45º à esquerda em função do sentido do vento até 45º para a direita, pelo que quando a embarcação se encontre dentro desta zona, esta deverá sempre arribar à bolina de forma a ter força suficiente na vela para fins de locomoção.

2.2.2.4. Navegação à bolina

A fim de um veleiro se deslocar em direção a um determinado objetivo tem de se garantir que a vela apanhe impulsão suficiente por parte do vento. Nos casos em que a orientação da proa do veleiro se encontre dentro da zona proibida de navegação há que adotar uma tática que permita a este deslocar-se fora desta a fim de evitar situações de inatividade, ou pior ainda, a navegação no sentido oposto ao pretendido.

A solução para estas situações será seguir uma trajetória aos ziguezagues em direção ao destino, sendo este tipo de navegação chamada de navegação à bolina [7], conforme apresentado na figura 2.5.

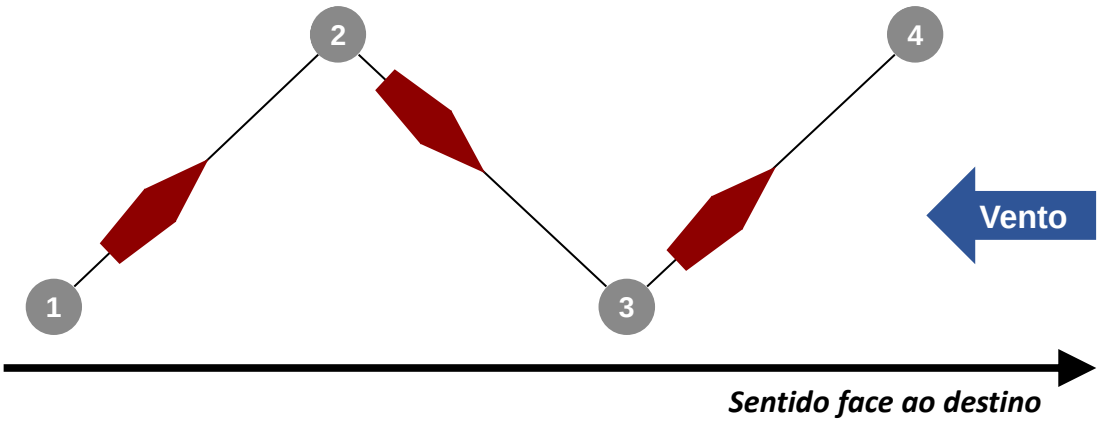


Figura 2.5: Navegação à bolina [6], [7].

2.3. Lógica difusa

Certas noções complexas da realidade que nos rodeia não permitem por vezes aplicar categorizações de caráter exato, sendo necessário outras formas menos rígidas de descrição de entidades.

2.3.1. Controlador Difuso

A estrutura de um controlador difuso consiste essencialmente em quatro componentes, cada um responsável por uma etapa do processo de controlo: base de conhecimento, fuzificação, motor de inferência e desfuzificação, conforme descrito na figura 2.6.

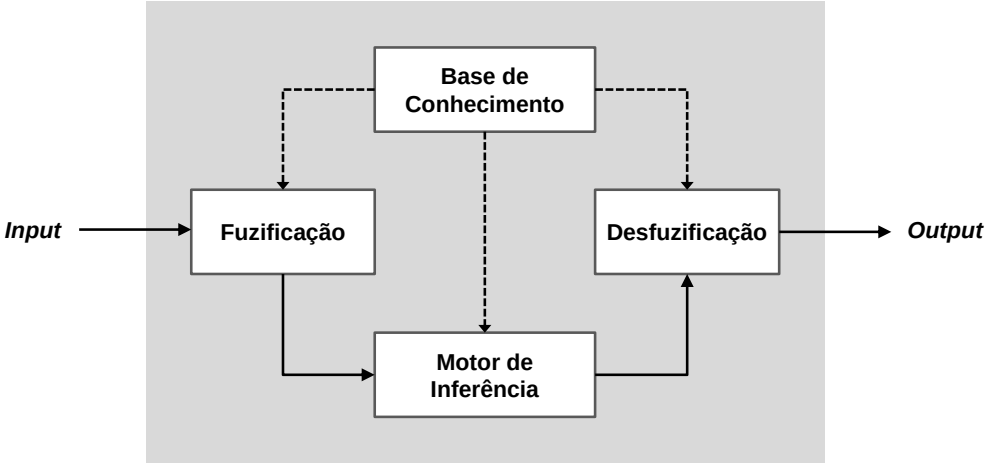


Figura 2.6: Controlador difuso [9].

2.3.1.1. Fuzificação

A fuzificação é um processo que consiste na intersecção entre um valor de entrada de um sistema (por exemplo, um sensor) e uma função pertença que abrange um intervalo de um certo universo de discurso (escala da variável de entrada), como descrito na figura 2.7.

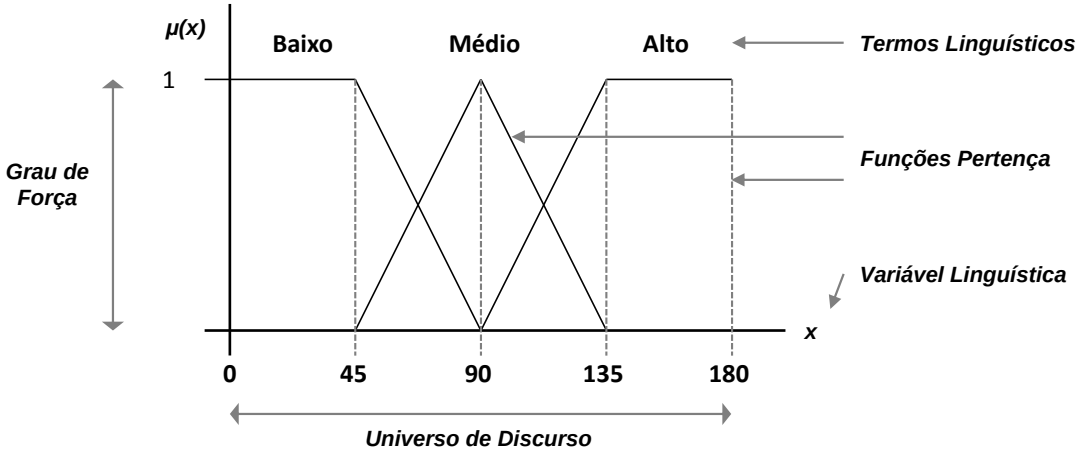


Figura 2.7: Exemplo genérico de um conjunto de funções pertença.

As funções pertença podem tomar diferentes formas tais como triângulos, trapézios, gaussianas, etc [9].

Os diferentes conjuntos de valores do universo de discurso, definidos pelas respetivas funções pertença, chamam-se termos linguísticos que são nada mais que classificações qualitativas definidas pelo criador.

O eixo vertical representa o grau de força ou grau de pertença, isto é, o valor que transita para a determinação do valor de saída em função do valor crespo de entrada do universo de discurso.

2.3.1.2. Base de conhecimento

A base de conhecimento consiste essencialmente em duas etapas realizadas por parte do seu criador.

A primeira trata da definição dos termos linguísticos que definem as funções pertença a cruzar com os valores de entrada, assim como na definição dos valores de saída [9].

A segunda consiste na definição das regras de inferência que determinam quais os consequentes (funções pertença de saída) resultantes dos antecedentes estabelecidos (funções pertença de entrada), seguindo um conjunto de regras do tipo "if-else" [9] conforme descrito no exemplo genérico 2.2.

IF *antecedente1* **operador** *antecedente2* **THEN** *consequente* (2.2)

Os operadores (de utilização opcional) correspondem a operações lógicas tais como "AND", "OR" ou "NOT", podendo também agregar múltiplos consequentes.

2.3.1.3. Decisão

Este componente diz respeito ao facto de poder haver mais que uma regra que dispare em simultâneo com consequências para a respetiva função pertença de saída [9]. É decidido então, pelo operador, qual o valor que transita para o processo de desfuzificação, de acordo com a figura 2.8.

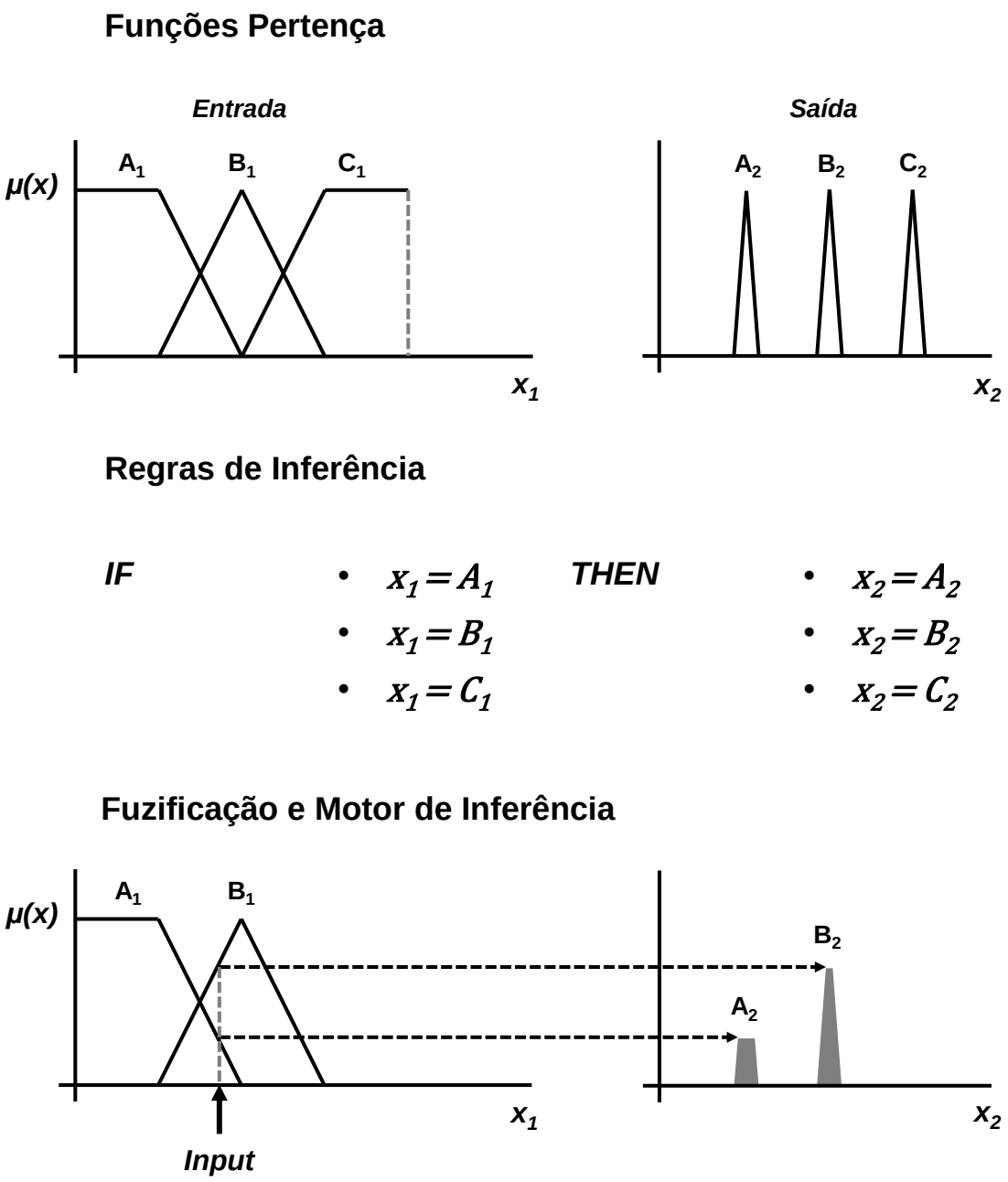


Figura 2.8: Exemplo genérico de um processo de fuzificação e do motor de inferência.

2.3.1.4. Desfuzificação

É o processo de transição do valor final definido pelas áreas resultantes do motor de inferência, definindo o método utilizado para se determinar o valor final da saída do sistema.

Este método pode utilizar diferentes abordagens de cálculo tais como o cálculo de centro da área ("COA") ou o método resultante da média dos máximos ("MOM") [9].

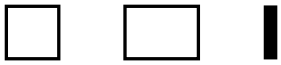
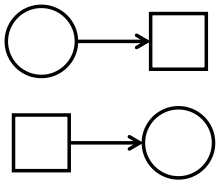
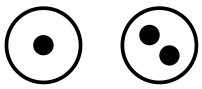
2.4. Redes de Petri

O sistema de controlo presente nesta dissertação utiliza uma máquina de estados em notação de rede de Petri. As redes de Petri (RdP) são um formalismo gráfico abstrato baseado numa generalização do conceito de máquinas de estado que permitem uma compreensão intuitiva do comportamento do sistema. Estas têm como vantagem permitir a elaboração de sistemas que verifiquem as seguintes características:

- **Concorrência ou paralelismo:** a possibilidade de dois ou mais conjuntos de estados e transições apresentarem dependência mútua ou correrem em simultâneo;
- **Sincronização:** necessidade de estabelecimento de relações de ordem no funcionamento do sistema a fim de certas transições dispararem em sincronia;
- **Exclusão mútua:** no caso de existir um recurso partilhado por vários sistemas ou subsistemas que não pode ser utilizado por mais de um deles em simultâneo.

As redes de Petri são constituídas essencialmente por 4 elementos principais: lugares, transições, arcos e marcas, de acordo com a tabela 2.1.

Tabela 2.1: Componentes de uma rede de Petri [11].

Nome	Símbolo	Descrição
Lugares		Componentes passivos que podem acumular, guardar ou mostrar objetos.
Transições		Componente ativo que pode produzir, consumir, transportar ou alterar objetos.
Arcos		Conector indicador do sentido do fluxo entre um lugar e uma transição ou vice-versa.
Marcas		Indicadores de objetos dos diversos lugares.

Uma transição só pode estar habilitada a disparar se todos os lugares de entrada conterem o número mínimo de marcas necessárias ao disparo da transição, dependendo do peso de cada arco, conforme está representado na figura 2.9.

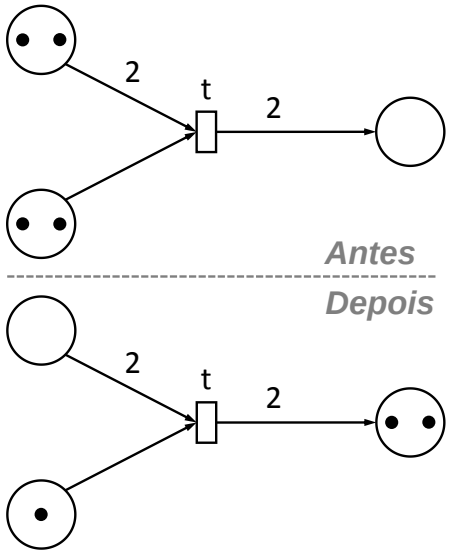


Figura 2.9: Exemplo de disparo de transição [12].

Capítulo 3

Linha de projetos eVentos

Este capítulo faz uma síntese das dissertações da linha de projetos eVentos que antecedem a presente dissertação. A linha de projetos eVentos é composta por três dissertações realizadas pela Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa que têm como intuito a definição e construção de uma estratégia de controlo para navegação autónoma de um pequeno veleiro.

A primeira dissertação desta linha de projetos, nomeada *eVentos 2 - Autonomous sailboat control* [2] estabelece o planeamento base a seguir, indicando possíveis componentes de hardware e software a utilizar assim como uma estrutura prévia de controlo a considerar.

A dissertação *eVentos 3 - Desenvolvimento de controlador difuso para navegação autónoma de veleiro* [3] é responsável pela implementação e teste de um controlador de navegação em condições de vento favorável, isto é, de definir uma estratégia de orientação face a uma etapa quando o veleiro se encontra fora da zona proibida de navegação.

A dissertação *eVentos 4 - Controlador para navegação autónoma de veleiro em modo de regata* [4] define uma estratégia de navegação em condições de vento desfavorável a fim de permitir o veleiro navegar à bolina face ao próximo objetivo quando este se encontra a navegar dentro da zona proibida.

Além das dissertações mencionadas existem algumas dissertações de caráter auxiliar que definem componentes e estratégias extra referentes a certos casos particulares de navegação. A dissertação *Desenvolvimento de um Sistema anticolisão para um veleiro com navegação autónoma* [13], [14] define métodos de deteção de obstáculos por utilização de uma câmara Pixy [15].

As dissertações *Prova de evasão de obstáculos em competição de veleiros com navegação autónoma* [16] e *Prova de varrimento de área em competição de veleiros com navegação autónoma* [17] definem estratégias orientadas a competições referentes ao evento internacional WRSC/IRSC (World Robotic Sailing Championship and International Robotic Sailing Conference), nomeadamente as provas de "collision avoidance" e "area scanning" da série de 2017 ocorrida em Horten, Noruega [18].

Sendo que a presente dissertação eVentos 5 tem como objetivo a construção de um sistema de controlo funcional capaz de permitir a um pequeno veleiro navegar de forma autónoma ao longo de um trajeto de etapas pré-definidas, quer em condições de vento favorável como desfavorável, consideraram-se as dissertações da linha eVentos como o foco principal do trabalho desenvolvido, a fim de se obter um sistema unificado que garanta o cumprimento de todos os objetivos propostos.

3.1. eVentos 2

A primeira dissertação da série de projetos eVentos apresentada foi a dissertação *eVentos 2 - Autonomous sailboat control* [2]. Nela especificam-se ferramentas de software e hardware propostas para a elaboração de um controlador de navegação autónoma para pequenos veleiros, assim como bases possíveis da estratégia de controlo a considerar em dissertações seguintes.

Apesar de não terem ocorrido quaisquer testes de navegação ou montagem e preparação de um veleiro de teste, esta não deixa de ser útil na definição dos passos a seguir a fim de se obter um modelo experimental funcional.

3.1.1. Hardware proposto

Não se tendo procedido à montagem completa de um sistema de controlo apto a navegação, a dissertação eVentos 2 limitou-se a propor o uso de vários componentes de hardware que pudessem ser úteis a projetos futuros.

3.1.1.1. Microcontrolador Arduino

A plataforma proposta para a implementação da estratégia de controlo foi um microcontrolador reconfigurável Arduino Mega 2560 [19]. Esta escolha deveu-se a várias propriedades que a caracterizam tais como o baixo custo, a propriedade "cross-platform" do IDE e pelas bibliotecas "open source" disponíveis para a elaboração de código [2].

3.1.1.2. Sensores propostos

Nesta dissertação propôs-se o uso de 4 sensores distintos:

- **Anenómetro e cata-vento:** sensores necessários à medição da velocidade e sentido do vento. Para tal recomendou-se um kit da Argent Data Systems [2], [20]. Este kit também conta com um medidor de pluviosidade apesar deste não ser necessário;
- **Bússola magnética:** foi também proposto uma bússola magnética CMPS10 [2], [21] que contem um magnetómetro a 3 eixos de forma a medir os ângulos de rotação referentes à embarcação, assim como um acelerómetro também a 3 eixos e um processador de 16 bits com uma tensão de alimentação de 3,6 a 5V, a 25 mA de corrente nominal;
- **GPS e antena:** recomendou-se o uso de um GPS EM-406a SiRF III [2], [22] da Global Sat [23] para fins de determinação da localização geográfica, comunicando com os satélites segundo as especificações NMEA [24].

3.1.2. Software

Nesta dissertação fez-se uso de apenas uma ferramenta de autoria externa, tendo sido esta utilizada para a elaboração de controladores difusos.

3.1.2.1. XFuzzy

Para fins de simplificação da geração de código utilizou-se a plataforma de desenvolvimento de sistemas de controlo difuso por inferência XFuzzy [25].

A razão desta escolha deve-se à capacidade desta de elaborar sistemas de controlo difuso complexos ao mesmo tempo que permite ao utilizador manipular conjuntos de funções pertença com facilidade [2].

3.1.3. Sistema de controlo proposto

A estrutura de controlo proposta parte de 4 sinais de entrada (figura 3.1) provenientes dos sensores GPS, bússola, cata-vento e anemómetro [2].

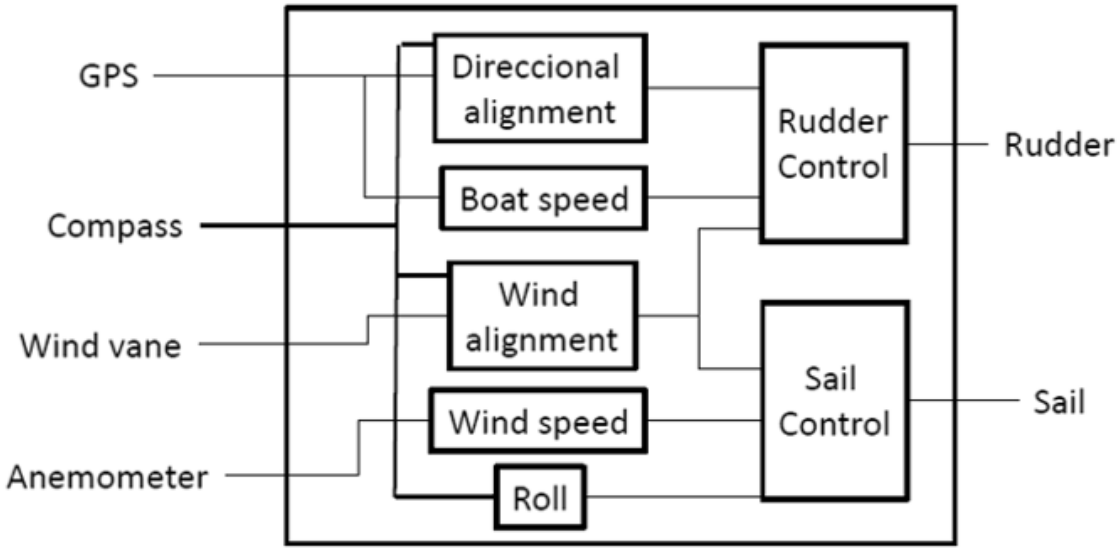


Figura 3.1: Variáveis usadas para a ação de controlo do leme e da vela [2].

Sobre estes é feito um pré-processamento a dois níveis: a determinação do alinhamento direcional (a posição angular do veleiro em função do destino) e a determinação do alinhamento do vento, que é no fundo a orientação angular do sentido do vento em função da proa da embarcação [2].

Após se obterem todas as variáveis necessárias, a ação de controlo é desempenhada por dois controladores difusos distintos. O controlador do leme recebe o alinhamento do vento assim como o alinhamento direcional e a velocidade do veleiro, de forma a determinar o valor a enviar para o servo do leme. O controlador da vela recebe o valor do "roll" assim como o alinhamento do vento e a sua velocidade, de forma a atuar no servo da vela [2].

3.1.3.1. Variáveis de controlo

Dos sinais obtidos pela leitura dos sensores são consideradas 5 variáveis a partir das quais se definem as regras de inferência dos controladores difusos [2]:

- **Alinhamento direcional:** determinará se a embarcação procede a uma viragem a bombordo ou estibordo consoante a direção do destino a atingir;
- **Velocidade da embarcação:** quanto mais rápida for a velocidade da embarcação, mais atenuada será a ação do leme a fim de prevenir possíveis situações de naufrágio;
- **Alinhamento do vento:** o alinhamento do vento face ao sentido da proa da embarcação determina se a vela é folgada, caso o vento incida na ré, ou caçada caso o vento incida na proa da embarcação;
- **Velocidade do vento:** tem um efeito na vela análogo ao efeito que a velocidade da embarcação tem no leme na medida em que atenuará o grau a que a vela será folgada mediante o seu valor;
- **Roll:** determina a rotação da embarcação no eixo proa-popa, ou seja, quanto mais inclinada esta se encontre, mais folgada será a vela.

3.1.3.2. Transição entre controlo manual e controlo automático

Admitindo a possibilidade de ocorrência de falhas por parte do controlador automático definiu-se uma estratégia de comutação que permite ao utilizador mudar a sua vontade entre os modos de controlo automático, deixando o controlador definir as ações da embarcação de forma autónoma, e o modo de controlo manual desempenhado pelo próprio utilizador com recurso a um controlador de rádio.

Para permitir esta comutação analisou-se uma amostra do sinal enviado para o controlo dos servos por parte do controlador manual pertencente ao kit do veleiro. Verificando a frequência do sinal emitido para vários períodos de amostragem definiram-se dois valores de incremento e decremento assim como dois limites superior e inferior a fim de definir a estratégia de comutação [2].

Caso o controlador detete o sinal a 5 V este incrementará uma variável um dado número de unidades e quando o valor excede o limite superior é feita a comutação para controlo manual. De modo oposto, a não deteção do sinal (0 V) resulta num decremento e quando a variável se encontre abaixo do limite inferior, o controlador retomará o modo automático [2], conforme indicado na figura 3.2.

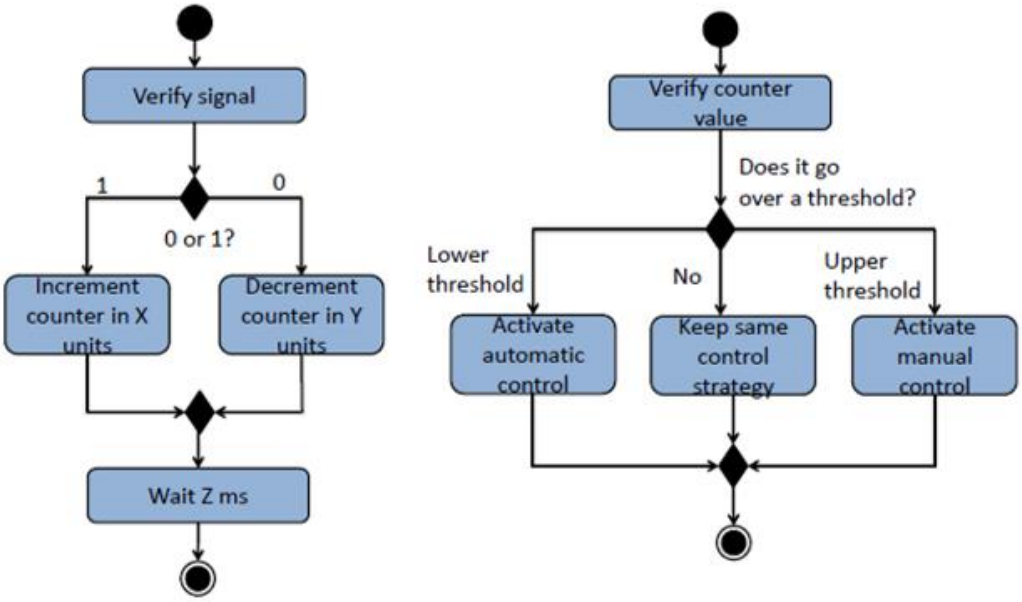


Figura 3.2: Diagramas de fluxo do contador *à esquerda* e de estratégia de transição *à direita* [2].

3.1.3.3. Estrutura de decisão do leme

Dada a impossibilidade de navegação na direção do vento considerou-se uma zona proibida de navegação com uma margem angular simétrica de 45º semelhante àquela já descrita na figura 2.4.

Com base nesta margem, propôs-se assim a divisão do sistema de controlo em três controladores: um controlador "Favorable Wind" para os casos em que o sentido do vento é favorável, i.e., o veleiro se encontre a navegar fora da zona proibida e dois controladores de bolina chamados "Tack Left" e "Tack Right" para os casos de vento desfavorável (dentro da zona proibida) [2].

Para permitir a navegação dentro da zona proibida, a máquina de estados propõe uma alternância entre os controladores de bolina baseada na velocidade do veleiro e na distância do veleiro ao objetivo, contudo não foram definidos valores concretos [2].

3.1.3.4. Aproximação de objetivo

Para determinar se o veleiro já atingiu o objetivo em direção ao qual deve navegar propôs-se duas estratégias de chegada distintas.

Numa delas considera-se uma área circular de raio pré-definido centrado no ponto de coordenadas de etapa a atingir. O objetivo é atingido caso o veleiro se encontre dentro dessa área, ou seja, caso a distância do veleiro ao destino seja inferior ao valor do raio da área circular [2].

Numa outra abordagem consideram-se três retas: uma reta de meta e duas retas de limite ortogonais a esta, separadas por uma dada distância. O objetivo é atingido caso o veleiro passe a linha de meta entre as linhas de limite [2].

3.1.4. Validação e análise de resultados

Apesar de não se ter implementado o sistema de controlo num veleiro, nem de se terem realizado testes num contexto de navegação, fizeram-se testes aos diferentes controladores utilizando a ferramenta de síntese do software XFuzzy, confirmando-se assim a resposta dos valores de saída do sistema a diversos valores de entrada [2].

3.1.5. Melhorias propostas

Na dissertação eVentos 2 são recomendadas duas melhorias a realizar em trabalhos futuros [2]:

- **Capacidade de navegar contra o vento:** a definição de uma estratégia de controlo que permita a embarcação navegar à bolina dentro da zona proibida;
- **Utilização de um router wireless:** implementação de um dispositivo para comunicação de longa distância para com a costa.

3.2. eVentos 3

A primeira dissertação a implementar e testar uma estratégia de controlo para contextos de navegação denomina-se *eVentos 3 - Desenvolvimento de controlador difuso para navegação autónoma de veleiro* [3] que teve como objetivo a elaboração de um controlador difuso para fins de navegação em condições de vento favorável, i.e., fora da zona proibida.

A dissertação eVentos 3, juntamente com a dissertação eVentos 4 [4], foram efetuadas de forma paralela e resultam de uma partição da estratégia de controlo em três níveis, tendo como objetivo final a realização de testes de navegação, quer do controlador eVentos 3, quer da ação conjunta de ambos os controladores.

3.2.1. Hardware

Para a realização desta dissertação escolheu-se, à semelhança do que foi proposto pela dissertação eVentos 2, uma plataforma reconfigurável Arduino Mega 2560 [19], a par com 3 sensores distintos [3].

Esta, mediante a estratégia de controlo, atuará sobre dois servos através do protocolo de comunicação PWM, sendo um deles responsável pelo controlo das velas e o outro pelo controlo do leme [3]. O hardware foi depois embarcado num modelo Naulantia [5].

Este sistema conta ainda com a possibilidade de operar em modo manual por uso de um controlador de rádio, assim como a capacidade de monitorização de navegação através de uma ligação a um computador, por utilização de um "Wi-Fi shield" [3]. Tem-se assim a arquitetura apresentada na figura 3.3.

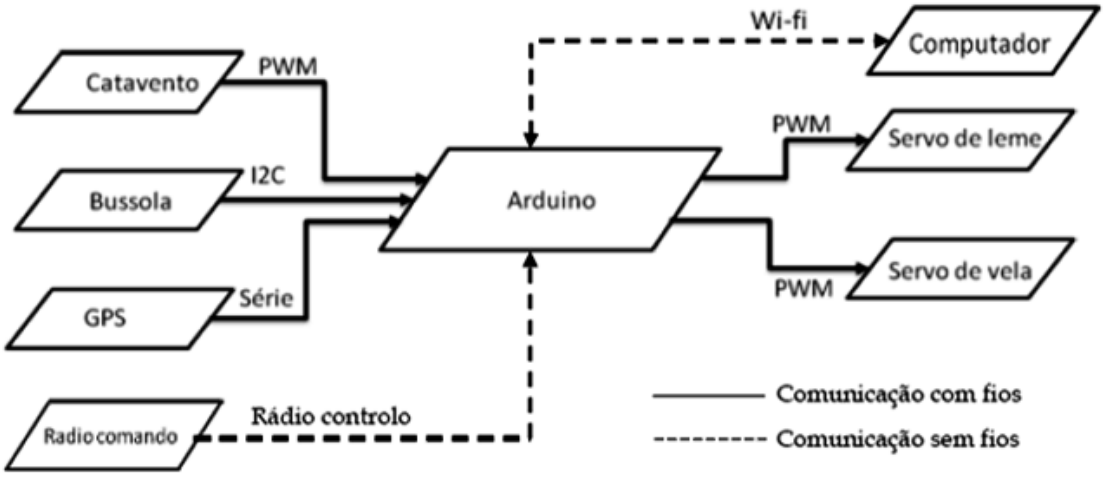


Figura 3.3: Arquitetura do hardware embutido [3].

3.2.1.1. Sensores

Nesta dissertação foram utilizados três sensores (figura 3.4) [3]:

- **Cata-vento:** Este sensor foi desenvolvido a partir de um circuito integrado AS5161 [26] que analisa a variação do campo magnético, sendo a estrutura conseguida através de uma peça impressa numa impressora 3D. Tem como função a determinação do sentido do vento, sendo a sua comunicação para com o Arduino feita em PWM;

- **GPS:** Para a determinação da localização do veleiro utilizou-se um GPS MT3329 [27] que comunica com os satélites via mensagens NMEA, com o intuito de obter as coordenadas atuais de latitude e longitude do veleiro. A comunicação com o Arduino é realizada por protocolo série, tendo também um led de sinalização de captação de sinal;

- **Bússola:** Para este projeto escolheu-se uma bússola magnética de modelo CMPS10, igual àquela proposta pela dissertação eVentos 2.

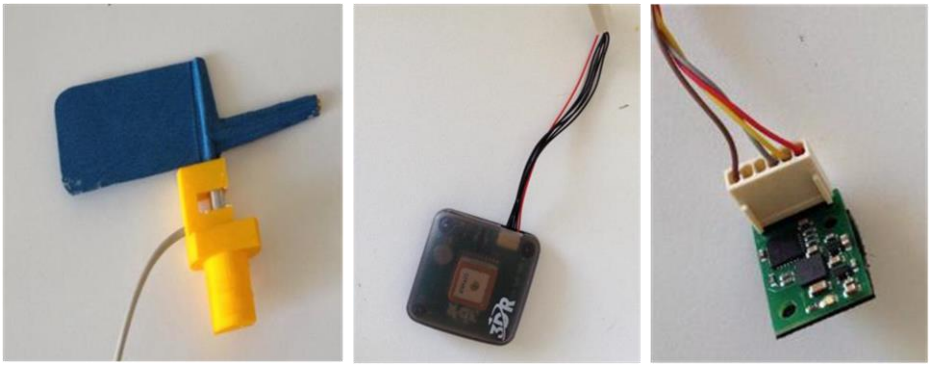


Figura 3.4: Cata-vento com sensor AS5161 incorporado [3], [26] *à esquerda*; GPS MT3329 [3], [27] *ao centro*; Bússola CMPS10 [3], [21] *à direita*.

3.2.1.2. Aquisição remota de dados e controlo manual

Além dos elementos base essenciais à ação de controlo foram também utilizados dois componentes de caráter auxiliar à navegação: um sistema de monitorização de dados para guardar os resultados de testes efetuados e um sistema de controlo manual [3], apresentados na figura 3.5.

Para a elaboração do sistema de monitorização de dados foi necessário a incorporação de um "Wi-Fi shield" [28] para permitir a troca de informação entre o computador e a plataforma Arduino. O "shield" comunica com o Arduino por um protocolo série e com o computador via Wi-Fi [3].

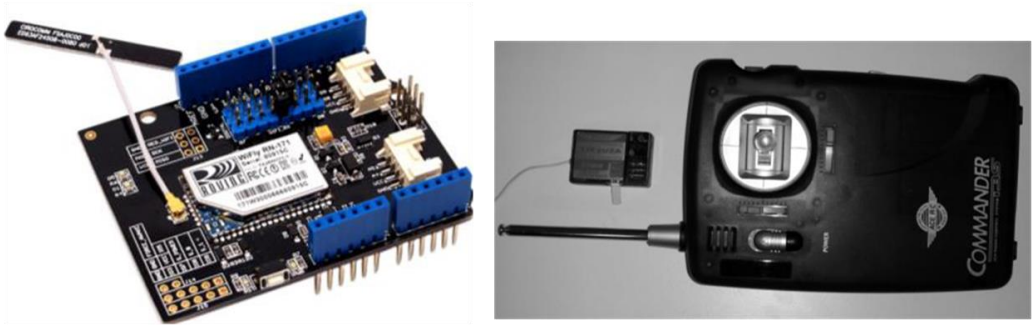


Figura 3.5: "Wi-Fi shield" [3], [28] à *esquerda*; Sistema de comando manual [3] à *direita*.

O controlo manual é realizado por intermédio de um controlador rádio juntamente com um recetor de três canais: dois atribuídos ao controlo do leme e da vela e um terceiro reservado à comutação entre os modos de navegação manual e autónoma [3].

3.2.2. Software

Para a elaboração da estratégia de controlo foi utilizada a plataforma XFuzzy para a construção de um sistema de controlo difuso, à semelhança do que foi proposto na dissertação eVentos 2. Além disso, foi criada uma aplicação de monitorização de dados.

3.2.2.1. Aplicação de gestão e monitorização de dados

Para a monitorização e armazenamento de dados de navegação realizou-se uma aplicação gráfica construída com linguagem Java (figura 3.6).

Esta aplicação teve como objetivo guardar os dados relativos à navegação, o envio de coordenadas de destino e a verificação do percurso do veleiro [3].

Para a comunicação entre esta aplicação e o Arduino desenvolveu-se um protocolo de "sockets UDP" para envio de datagramas compostos pelos valores necessários à monitorização do veleiro [3].

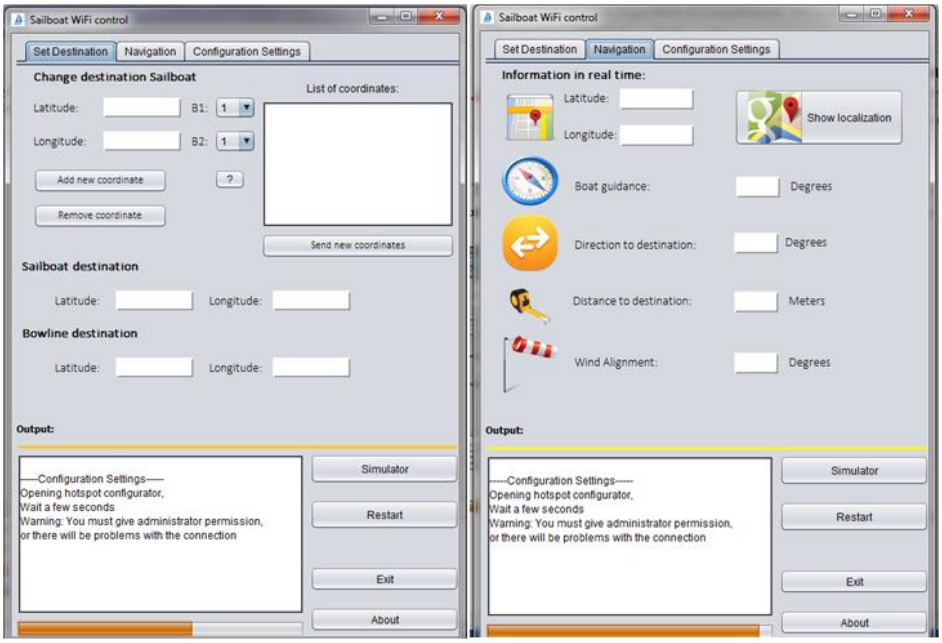


Figura 3.6: Aplicação de aquisição de valores [3].

3.2.3. Sistema de controlo

O sistema desenvolvido baseia-se numa partição da estrutura de controlo em três níveis, conforme mostrado na figura 3.7. Esta abordagem resulta na separação dos problemas inerentes a cada componente para uma análise mais focada em cada módulo individual [3], [29].

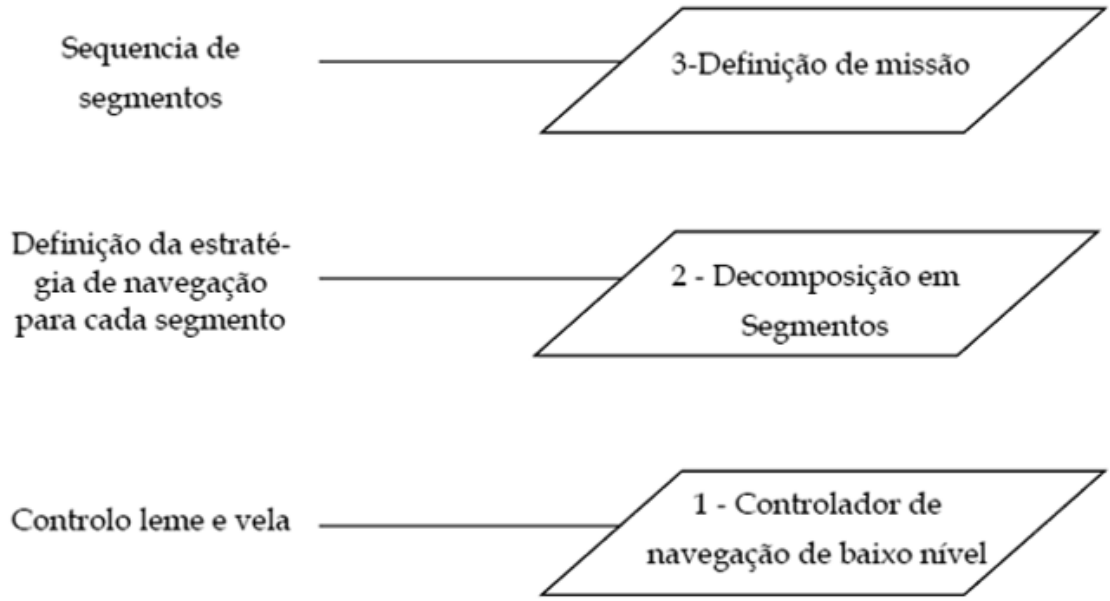


Figura 3.7: Estrutura hierárquica do controlador de eVentos 3 [3].

Cada segmento é responsável por uma porção diferente da estratégia de controlo [3]:

- **Nível 3 – Definição de missão:** este nível diz respeito à especificação de características do modo regata, ou seja, a capacidade de definir várias etapas de percurso;
- **Nível 2 – Decomposição em segmentos:** nível responsável pelo controlo quando o vento se encontra desfavorável, isto é, quando a embarcação se encontra a navegar na zona proibida, sendo necessário criar objetivos intermédios. Este, mais o nível 3, é tratado na dissertação eVentos 4 [4];
- **Nível 1 – Controlador de navegação de baixo nível:** tem como objetivo o controlo do leme e vela em situação de vento favorável à navegação.

3.2.3.1. Nível 1

O nível 1 do controlador (controlador de navegação de baixo nível [3]) diz respeito ao controlo autónomo da embarcação em condições de vento favorável. Para o efeito são lidos os valores dados pelos três sensores: o sentido do vento (cata-vento), a orientação do veleiro face ao Norte (bússola magnética) e a sua localização geográfica (GPS).

De seguida ocorre um pré-processamento feito a partir dos valores da bússola e do GPS, determinando-se o ângulo de orientação do destino que é a orientação da proa do veleiro face ao objetivo a atingir, sendo este calculado a partir do vetor direção ao destino (vetor que parte da embarcação até ao ponto de destino) e do valor da bússola [3].

O ângulo de orientação do destino, juntamente com a orientação do vento dado pelo cata-vento, constituem as variáveis de saída do pré-processamento [3], conforme descrito na figura 3.8.

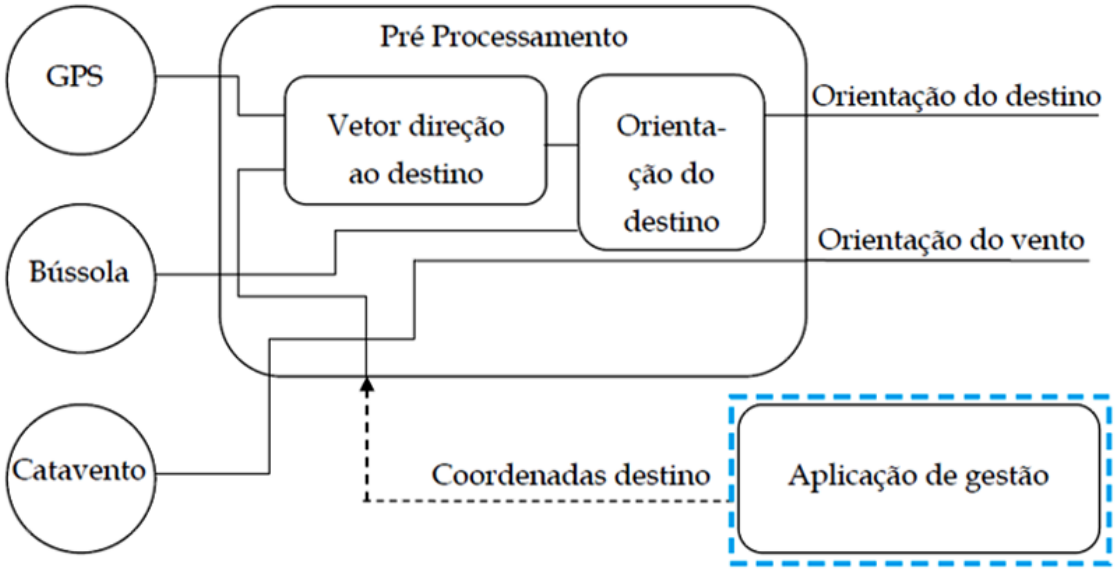


Figura 3.8: Arquitetura do software embutido [3].

O próximo passo foi a elaboração do controlador difuso em ambiente XFuzzy, a fim de definir as características do motor de inferência que o constitui.

O controlador tem duas variáveis de entrada, sendo estas o ângulo de orientação do destino e o ângulo de orientação do vento. Como variáveis de saída tem também dois valores a enviar aos atuadores do veleiro, sendo estes um servo responsável pelo controlo do leme e um servo responsável pelo controlo das velas [3], de acordo com a figura 3.9.

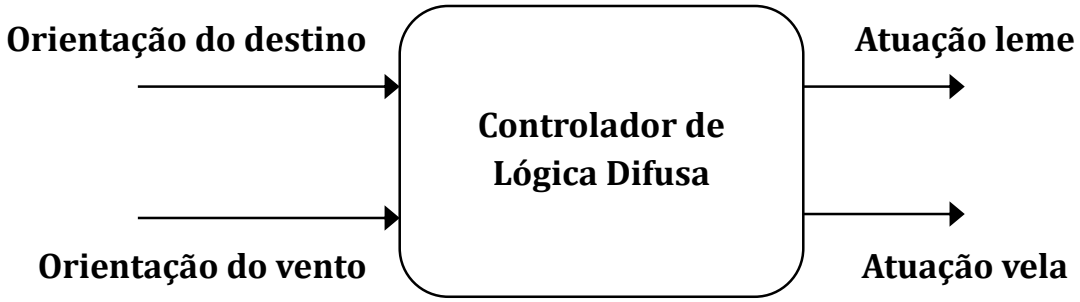


Figura 3.9: Entradas e saídas do controlador, segundo eVentos 3 [3].

Para cada uma destas variáveis definiu-se um conjunto de funções pertença, atribuindo a cada função um termo linguístico distinto. No fim definiram-se as regras de inferência do leme e da vela, abrangendo assim todas as situações possíveis de navegação [3].

3.2.4. Validação e análise de resultados

Numa primeira fase procedeu-se a um teste de navegação num pequeno lago (lago do Parque da Paz em Almada) a fim de verificar a eficácia do controlador em levar a embarcação de um ponto a outro em trajeto simples [3].

Feita uma revisão dos dados de navegação obtidos pela aplicação de monitorização procedeu-se à adição de uma nova função pertença (função 4 de termo linguístico centrado) até ai não existente, o que resultou numa segunda fase de teste com melhoria de resultados [3].

3.2.5. Melhorias propostas

Nesta dissertação foi feita a proposta de 4 melhorias possíveis para implementação em trabalhos futuros [3]:

- **Controlo do roll:** Um dos problemas verificados no decurso dos testes foi a entrada de água na embarcação causada pela inclinação excessiva desta devido ao vento. Uma possível solução para este problema é a adição de um conjunto de funções pertença extra ao controlador difuso, tendo em conta a leitura do "roll" feito pela bússola;

- **Análise de correntes marítimas:** A fim de lidar com a corrente marítima foi proposto um sensor de medição de corrente que juntamente com a velocidade do veleiro em função à terra, medida por GPS, permite determinar a velocidade do veleiro em relação à água;

- **Bomba de água:** A fim de drenar o interior do casco, expulsando a água que entra nos orifícios pelos quais saem as cordas de ligação das velas ao servo, propõe-se o uso de uma bomba de água;

- **Autonomia energética:** Sendo a alimentação do veleiro realizada com recurso a pilhas elétricas propôs-se o uso de painéis fotovoltaicos.

3.3. eVentos 4

A próxima dissertação da linha eVentos intitula-se *eVentos 4 - Controlador para navegação autónoma de veleiro em modo de regata* [4], sendo esta a antecessora imediata da dissertação presente.

Esta dissertação, feita em paralelo com a dissertação eVentos 3, teve como objetivo a elaboração de uma estratégia de controlo referente aos níveis 2 e 3 da estrutura indicada na figura 3.7 a fim de permitir a navegação do veleiro à bolina e em modo de regata, ou seja, que torne possível a este navegar contra o vento ao longo de um conjunto de etapas pré-definidas.

3.3.1. Hardware

O hardware utilizado nesta dissertação é o mesmo que foi utilizado na dissertação eVentos 3, tendo-se feito uso dos mesmos sensores (cata-vento [26], bússola magnética [21] e GPS [27]), a mesma plataforma reconfigurável Arduino Mega 2560 [19] e o mesmo modelo de embarcação [4], [5].

3.3.2. Software

Nesta dissertação utilizou-se, à semelhança da dissertação eVentos 3, a mesma plataforma de monitorização de resultados por transmissão de dados UDP via Wi-Fi.

Além disso fez-se uso da ferramenta de geração de código IOPT-Tools [30] para a elaboração e geração de código de redes de Petri.

3.3.3. Sistema de controlo

A dissertação eVentos 4 complementa o trabalho desenvolvido na dissertação eVentos 3 pela definição dos níveis 2 e 3 do sistema de controlo [4].

3.3.3.1. Nível 2

O nível 2 (decomposição em segmentos) é o nível de controlo responsável por permitir a navegação à bolina. Para tal define-se um corredor geométrico de navegação dentro do qual o veleiro tomará uma trajetória em ziguezague ao longo de várias etapas intermédias, a fim de garantir a aproximação para com o destino final. Este é usado em conjunto com uma RdP a fim de coordenar a ação do controlador entre os diferentes estados de navegação [4].

O corredor de navegação, definido desde um ponto de origem até ao ponto de destino, é composto por um total de 6 retas e duas balizas [4], conforme apresentado na figura 3.10.

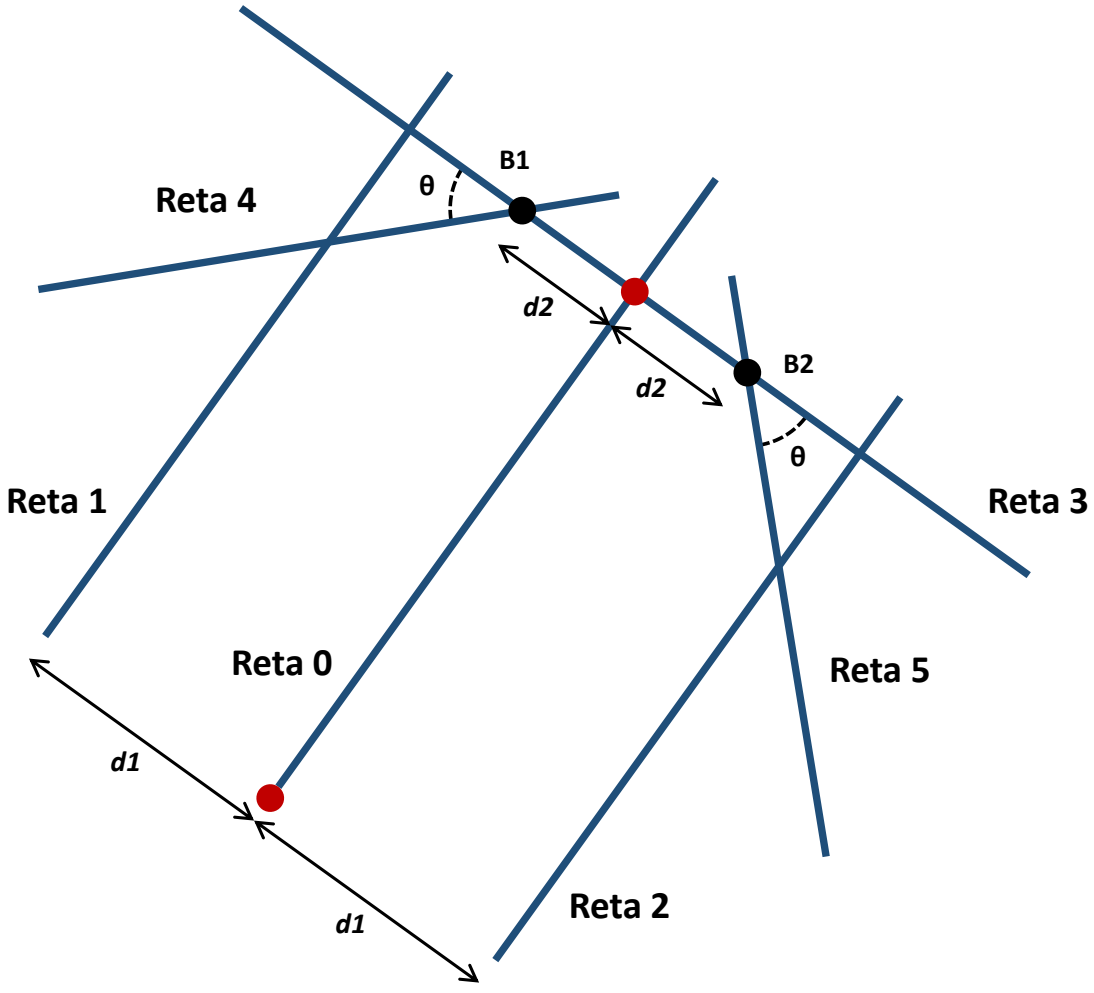


Figura 3.10: Constituintes do corredor segundo eVentos 4 [4].

Primeiro é calculada a reta 0 que une os pontos de origem e destino do corredor. De seguida são calculadas as retas 1 e 2 limitadoras do corredor, sendo estas equidistantes da reta 0 por adição/subtração de um offset d1 [4].

De seguida são calculadas duas balizas de aproximação (pontos B1 e B2) equidistantes do ponto de destino por uma distância d2. Estas encontram-se ao longo de uma reta ortogonal à reta 0 que passa no ponto de destino, denominada reta 3 [4].

No fim, a partir de cálculo tangencial, são determinadas as retas 4 e 5 a partir de uma rotação de 45º para com a reta 3. Estas definem um estreitamento a fim de levar o veleiro a convergir para com o objetivo.

Depois de se ter feito o cálculo do corredor há a necessidade do controlador saber se o vento está numa situação desfavorável. Para tal definiram-se três vetores (figura 3.11): o vetor $V_{alinhado}$ que representa a orientação do veleiro para com o destino, o vetor V_{frente} que define a orientação da proa da embarcação e o vetor V_{vento} relacionado com o sentido do vento em relação à proa. Além destes utiliza-se também o vetor $V_{referência}$ que é o vetor que une os pontos de origem e destino do corredor [4].

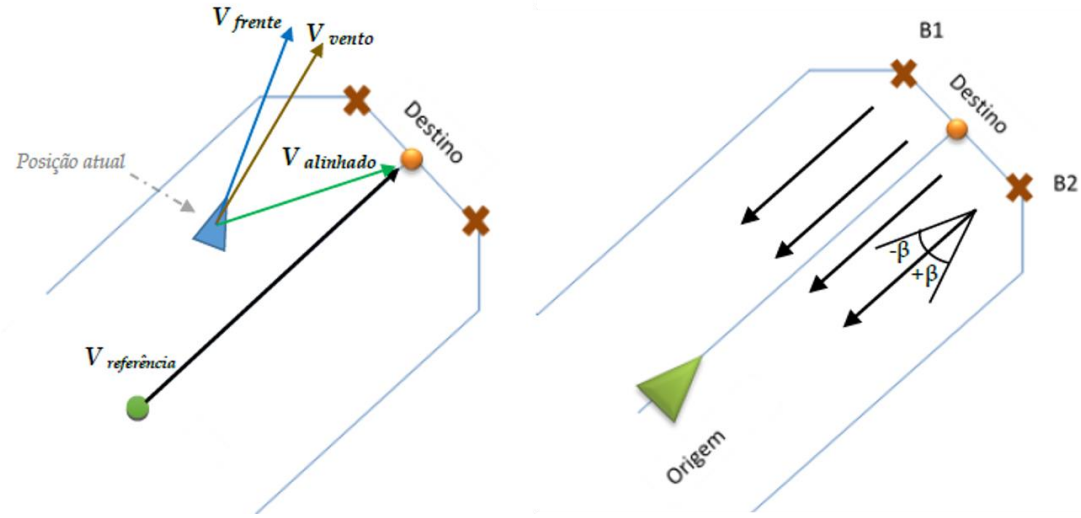


Figura 3.11: Representação dos vetores para detetar vento desfavorável *à esquerda*; Ângulo de bolina *à direita* [4].

Depois da definição dos vetores verifica-se o ângulo entre V_{vento} e $V_{referência}$. Se este estiver dentro do intervalo de -β a β que define a zona proibida considera-se que a zona do corredor se encontra com vento desfavorável. Assim que esta se verifique é determinada então uma coordenada intermédia a fim de o veleiro se dirigir até ela à bolina, sendo esta calculada pela translação da reta 0 até ao ponto atual do veleiro, seguido de uma rotação e posterior interseção com a reta 3 [4].

Para o veleiro ser capaz de realizar a trajetória aos ziguezagues à bolina em situação de vento desfavorável foi necessário a definição de zonas de navegação dentro do corredor. Dadas as especificidades do cálculo também se definiram dois cenários distintos pois a posição relativa das retas umas às outras ao longo do eixo vertical varia mediante o valor inferior ou superior de longitude da coordenada de destino face à longitude do ponto de origem [4], de acordo com a figura 3.12.

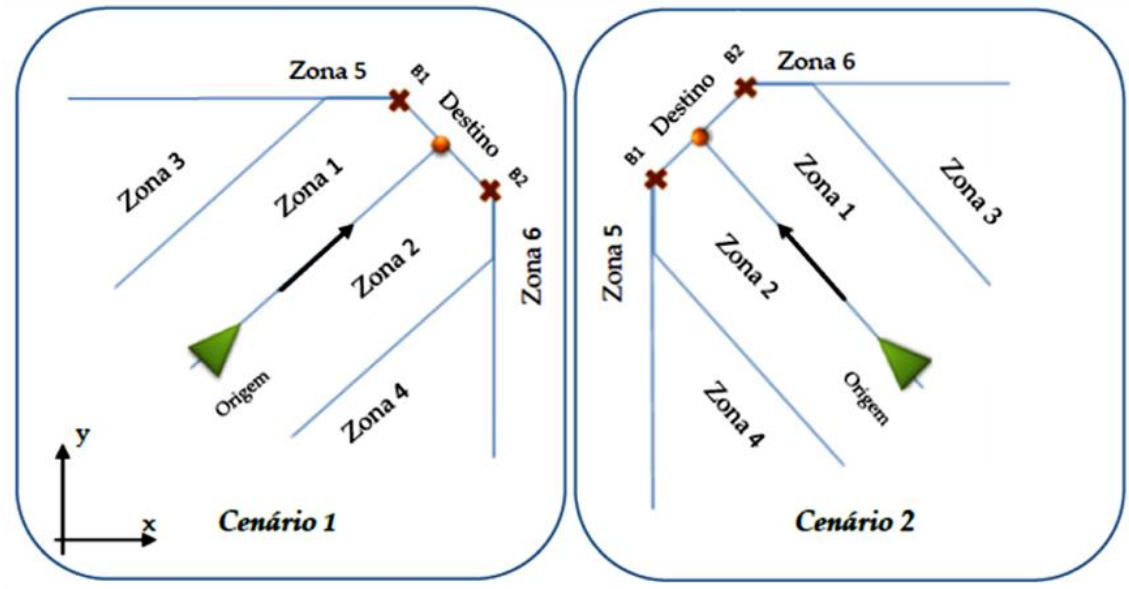


Figura 3.12: Zonas e cenários de navegação [4].

A fim de coordenar as ações entre os diferentes níveis do controlador presentes nas dissertações eVentos 3 e eVentos 4 implementou-se uma RdP.

Esta rede é composta por três lugares associados aos diferentes estados de navegação (figura 3.13): o estado *Normal* quando a navegação não se encontra à bolina (referente ao nível 1 do controlador) e dois estados de navegação à bolina referentes ao nível 2 chamados *Virar a bombordo* e *Virar a estibordo* [4].

Os estados de navegação à bolina determinam assim, caso a embarcação se encontre a navegar contra o vento, se a coordenada intermédia a calcular se encontra a bombordo ou a estibordo. O disparo das transições da RdP depende do estado de bolina, do cenário de navegação e da zona atual do corredor onde o veleiro se encontra [4].

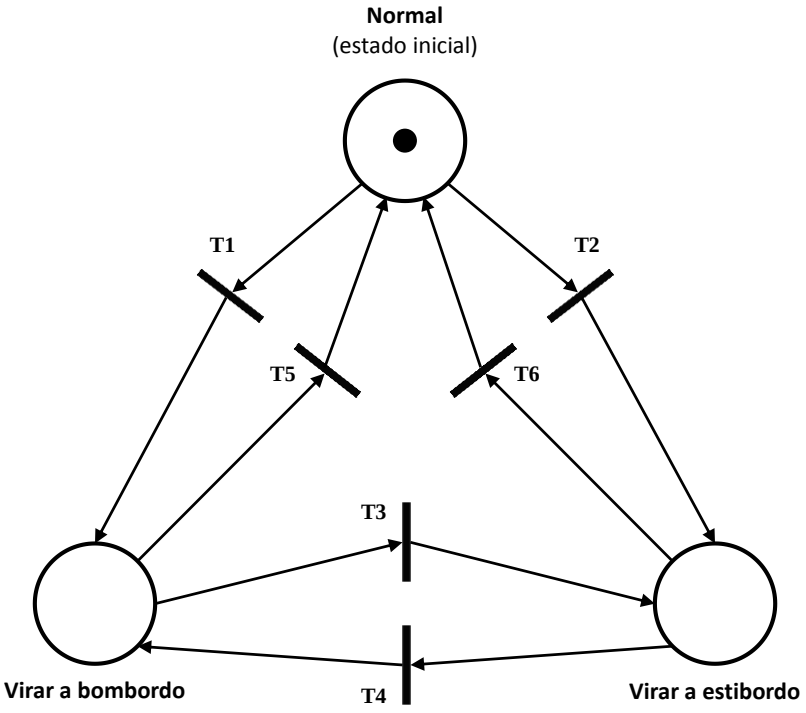


Figura 3.13: Máquina de estados em notação rede de Petri segundo eVentos 4 [4].

3.3.3.2. Nível 3

O nível 3 (definição de missão) tem como objetivo comunicar com o nível 2 a fim de saber quando o objetivo atual foi atingido, de modo a fornecer as coordenadas do próximo objetivo [4].

O nível 3 é essencialmente composto por dois blocos [4]:

- **Base de dados:** lista ordenada da sequência de coordenadas a atingir;
- **Seleciona segmento:** tem como entrada a variável *Objetivo atingido* e como saída a variável *Objetivo* que é entregue ao nível 2.

3.3.4. Validação e análise de resultados

Para o teste do controlador desenvolvido fizeram-se dois conjuntos de testes: um deles em contexto laboratorial e outro em condições exteriores [4].

Para os testes laboratoriais procedeu-se ao teste do bloco *Calcula Geometria*, detentor dos métodos referentes ao nível 2 do controlador [4]. A partir da utilização de uma aplicação de simulação e de três pares de coordenadas definidas com recurso ao Google Maps [31] correspondentes aos pontos de localização do veleiro, origem e destino do corredor verificou-se a zona e coordenada intermédia resultantes por mudança do ponto de localização [4]. Esta observação de resultados foi efetuada com recurso à plataforma online de desenho gráfico desmos [32].

Para os testes fora de contexto laboratorial começou-se por testar mais uma vez a capacidade de deteção de bolina e a coordenada intermédia, resultantes a partir de leituras feitas no exterior [4].

No fim foram realizados testes de navegação. Para testar a navegação em condições favoráveis realizou-se o teste descrito em eVentos 3 (secção 3.2.4), assim como alguns testes realizados no Clube Náutico de Almada [4], [33].

O último teste de navegação consistiu na participação na prova de regata Robotics Exercise 2015 (REX 2015) desenvolvida pelo Centro de Investigação Naval da Marinha Portuguesa (CINAV) [34], localizada no canal do arsenal do Alfeite. Infelizmente, devido a condições adversas provenientes da corrente marítima sentida e da entrada de água no casco do veleiro, este teste veio-se a revelar inconclusivo [4].

3.3.5. Melhorias propostas

Nesta dissertação propôs-se duas melhorias para projetos futuros [4]:

- **Deteção de obstáculos:** A fim de evitar colisões propõe-se a utilização de um sensor para deteção de obstáculos, tais como sensores infravermelhos e sensores ultrassónicos [35].
- **Autonomia energética:** à semelhança da dissertação eVentos 3 propôs-se a utilização de painéis fotovoltaicos.

Capítulo 4

Hardware e Software

Este capítulo apresenta uma listagem dos componentes de hardware e software utilizados na elaboração do controlador, descrevendo as características da plataforma de controlo, sensores, atuadores, os módulos de comunicação por radiofrequência e de controlo manual assim como a menção das plataformas de desenvolvimento utilizadas para a elaboração e geração de código de controlo.

4.1. Hardware

Para a implementação do sistema de controlo utilizou-se a mesma plataforma reconfigurável Arduino utilizada nas dissertações eVentos 3 e eVentos 4 assim como três sensores da mesma natureza (bússola magnética, catavento e GPS), dois pares de módulos de transmissão por radiofrequência e um "joystick shield" para fins de controlo manual.

4.1.1. Microcontrolador Arduino

A plataforma utilizada para a implementação da estratégia de controlo foi um microcontrolador reconfigurável Arduino Mega 2560 [19], apresentado na figura 4.1. Esta é uma plataforma baseada no chip ATmega2560 da atmel [36] que conta com 54 pinos digitais (15 dos quais aptos a leitura de protocolo PWM), 16 pinos analógicos, 4 canais de comunicação série e um par de pinos para utilização de protocolo de comunicação I2C.

A alimentação pode ser garantida ou pelo socket de ligação USB (também usado para o carregamento do código), ou pelo socket "Wall-Wart" com um adaptador AC/DC de 5V, ou pelos pinos V_{in} e GND com uma tensão entre 7 e 12 volts. Os pinos 5V também podem ser utilizados para o efeito desde que haja a garantia de uma tensão regulada.

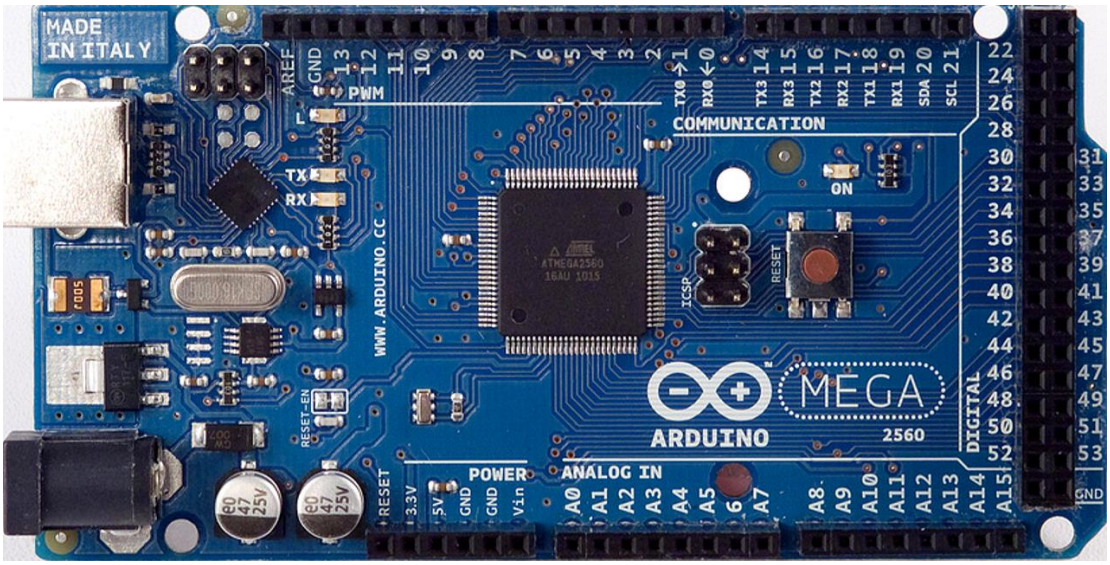


Figura 4.1: Microcontrolador Arduino Mega 2560 [37].

A programação desta plataforma é realizada em linguagem C/C++ com recurso a bibliotecas "open source", sendo utilizado para o efeito um IDE próprio da Arduino [38].

A tabela 4.1 sumariza algumas especificações adicionais relativas a este microcontrolador.

Tabela 4.1: Especificações da plataforma Arduino Mega 2560 [19].

Microcontroller	ATmega2560
Operating Voltage	*5V*
Input Voltage (recommended)	*7-12V*
Input Voltage (limit)	*6-20V*
Digital I/O Pins	*54 (of which 15 provide PWM output)*
Analog Input Pins	*16*
DC Current per I/O Pin	*20 mA*
DC Current for 3.3V Pin	*50 mA*
Flash Memory	*256 KB of which 8 KB used by bootloader*
SRAM	*8 KB*
EEPROM	*4 KB*
Clock Speed	*16 MHz*
Length	*101.52 mm*
Width	*53.3 mm*
Weight	*37 g*

4.1.2. Sensores e atuadores

O sistema de controlo de navegação conta com três sensores de entrada (cata-vento, bússola magnética e GPS), dois atuadores de saída (servos) e 5 componentes de caráter auxiliar (1 "joystick" e 4 módulos de comunicação RF).

4.1.2.1. Cata-vento

O cata-vento é constituído por um magnetómetro inserido numa estrutura de plástico sobre a qual assenta uma superfície de cartolina em forma de pá montada num eixo rotacional com um íman (figura 4.2), a fim de detetar o sentido do vento. De acordo com a posição atual da pá, o magnetómetro envia um sinal por protocolo PWM no "duty cycle" respetivo.

Este componente foi o mesmo utilizado nas dissertações eVentos 3 e eVentos 4, consistindo num magnetómetro de modelo AS5161 [26] inserido dentro da estrutura.

O valor correspondente de acordo com as especificações de fábrica insere-se num intervalo entre 0 e 4095 (12 bits) porém, devido a desgastes ocorridos com o tempo e ao desalinhar da pá de cartolina para com o eixo, convém fazer uma calibração antes de cada utilização para determinar os valores correspondentes aos extremos da escala.

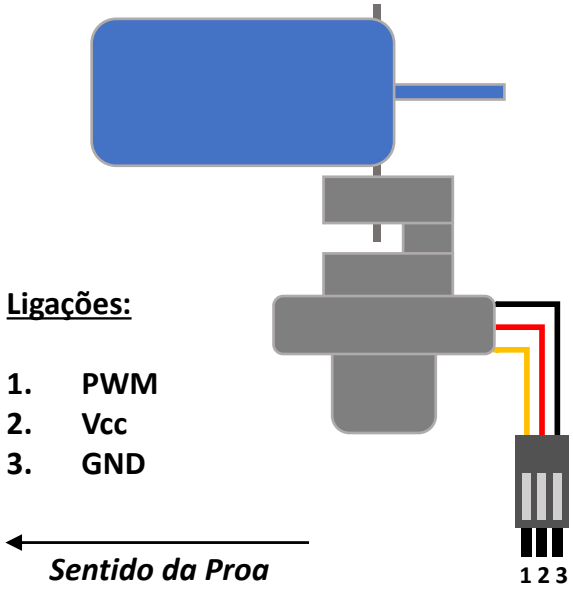


Figura 4.2: Esquemático de ligações do cata-vento.

4.1.2.2. Bússola magnética

A bússola magnética serve para medir a orientação angular face ao Norte magnético ("bearing") assim como os valores correspondentes aos movimentos rotacionais "pitch" e "roll" de acordo com a sua inclinação corrente. Para efeitos de controlo de navegação, apenas o "roll" e o "bearing" serão tidos em conta.

O modelo escolhido para o efeito, à semelhança das dissertações eVentos 3 e eVentos 4, foi um bússola CMPS10 [21]. Este modelo permite a comunicação em três modos possíveis, cada qual com um protocolo distinto, sendo estes PWM, série e I2C. Para a comunicação com o Arduino detentor do código de controlo escolheu-se o modo I2C (pinos de acordo com a figura 4.3) que consiste numa relação "master-slave" com o Arduino no papel de "master".

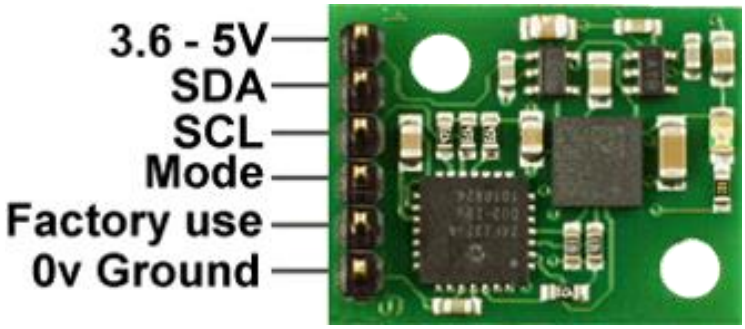


Figura 4.3: Esquema de ligações da bússola CMPS10 em modo I2C [21].

4.1.2.3. GPS

O GPS é o módulo sensorial responsável pela leitura das coordenadas geográficas de acordo com a sua localização corrente por intermédio de comunicação via satélite. Para fins de teste de controlo utilizou-se um GPS de modelo MT3339 [39] embutido num sistema Ultimate GPS da Adafruit [40] (figura 4.4), sendo este utilizado com uma biblioteca auxiliar do próprio fabricante [41].

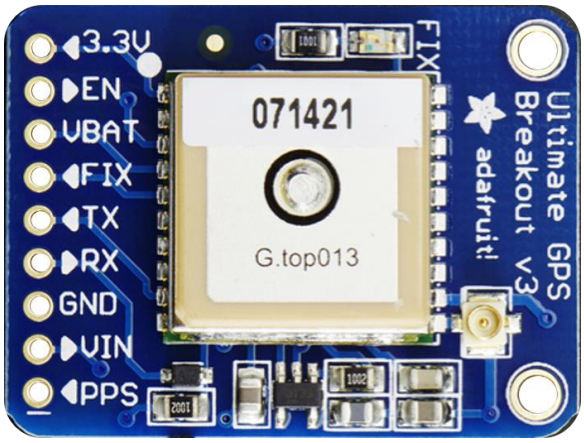


Figura 4.4: Sistema embutido Ultimate GPS da Adafruit [40].

Este dispositivo comunica com os satélites de geolocalização utilizando o formato de especificação de dados por "string" NMEA 0183 [24], [42] e com a plataforma Arduino utilizando o protocolo série.

Devido à dependência de satélites para a determinação da sua posição, não se recomenda o seu uso em ambientes fechados, devendo-se posicionar o sistema longe de edifícios que possam interferir na obtenção de sinal.

Outro ponto a ter em conta é que apesar de o modelo mencionado ter uma precisão dada pelos fabricantes de 1.8 metros, na prática é possível obter desvios de maior valor.

Com a necessidade de se obter uma conversão de metros para graus decimais de coordenada geográfica utilizou-se como referência o sistema geodésico americano WGS 84 [43]. Como os testes se realizam em território português, onde as coordenadas apresentam valores relativamente baixos de latitude, considerou-se para fins de cálculo das distâncias uma aproximação feita a partir da divisão do perímetro do equador por 360 (escala longitudinal), determinando-se assim que um grau de coordenada geográfica equivale a 111.319,5 metros.

4.1.2.4. Servos

O modelo Naulantia está dotado de dois servos electromecânicos: um servo HS-785HB [44] responsável pelo caçar/folgar da vela e um servo ZS-S2113 [45] responsável pelo controlo do leme, ambos apresentados na figura 4.5. Estes comunicam com a plataforma Arduino por utilização de protocolo PWM.

Figura 4.5: Servo HS-785HB [44] *à esquerda*; Servo ZS-S2113 [45] *à direita*.

Sendo que tanto a maneira como as cordas da vela são enroladas como a maneira que seja feita a ligação do leme sofrem alterações ao longo do seu uso, recomenda-se uma calibração prévia dos limites dos valores a enviar aos servos.

4.1.3. Módulos de comunicação rádio

Para garantir a capacidade do controlador de receber e enviar dados por radiofrequência, a fim de interagir com dispositivos periféricos distantes, utilizaram-se dois pares de módulos de comunicação rádio APC220 [46].

Estes módulos, apresentados na figura 4.6, comunicam por utilização de protocolo série, quer com o Arduino por ligação direta, quer com um computador por intermédio de um conversor TTL/USB, tendo um alcance máximo de comunicação de 1200 metros.

Figura 4.6: Par de módulos de comunicação rádio APC220 e conversor TTL/USB [47].

4.1.4. Joystick shield

O controlo manual do veleiro é assegurado por um joystick shield ITEAD [48] para Arduino (figura 4.7) provido de 6 botões e um manípulo analógico de dois eixos, cada qual com uma precisão de 10 bits.

Este dispositivo é usado em agregado com uma segunda plataforma Arduino que contem o código referente ao sistema de controlo manual do veleiro.

Figura 4.7: Joystick shield ITEAD [48].

4.2. Software

Para construir o controlador proposto utilizaram-se dois IDE's para geração de código e duas plataformas de auxílio à criação de estratégias de controlo.

O código Arduino foi criado utilizando um IDE do próprio Arduino mencionado na secção 4.1.1. Além deste utilizou-se o Netbeans IDE [49] para elaboração do código Java referente à aplicação de monitorização de resultados, tendo-se também utilizado este em conjunto com um compilador MinGW [50] para proceder a testes de componentes individuais do sistema com recurso a "scripts" em linguagem C++.

Para a elaboração do controlador difuso e máquina de estados fez-se uso do software XFuzzy e da plataforma online de construção de redes de Petri IOPT-Tools.

4.2.1. XFuzzy

Para fins de simplificação da geração de código utilizou-se a plataforma de desenvolvimento de sistemas de controlo difuso por inferência XFuzzy [25], cujo GUI se pode ver na figura 4.8.

O software XFuzzy é uma plataforma de desenvolvimento gráfico da autoria do Instituto de Microelectrónica de Sevilha que permite a elaboração de forma simples e intuitiva de sistemas de controlo difuso, proporcionando ferramentas de síntese para análise gráfica de resultados assim como opções de geração de código referentes aos sistemas desenvolvidos em linguagens C, C++, VHDL e Sysgen.

Dado que a plataforma Arduino utilizada é também programada em linguagem C/C++, é possível então utilizar o software XFuzzy para a criação de bibliotecas de controlo a integrar no código do controlador.

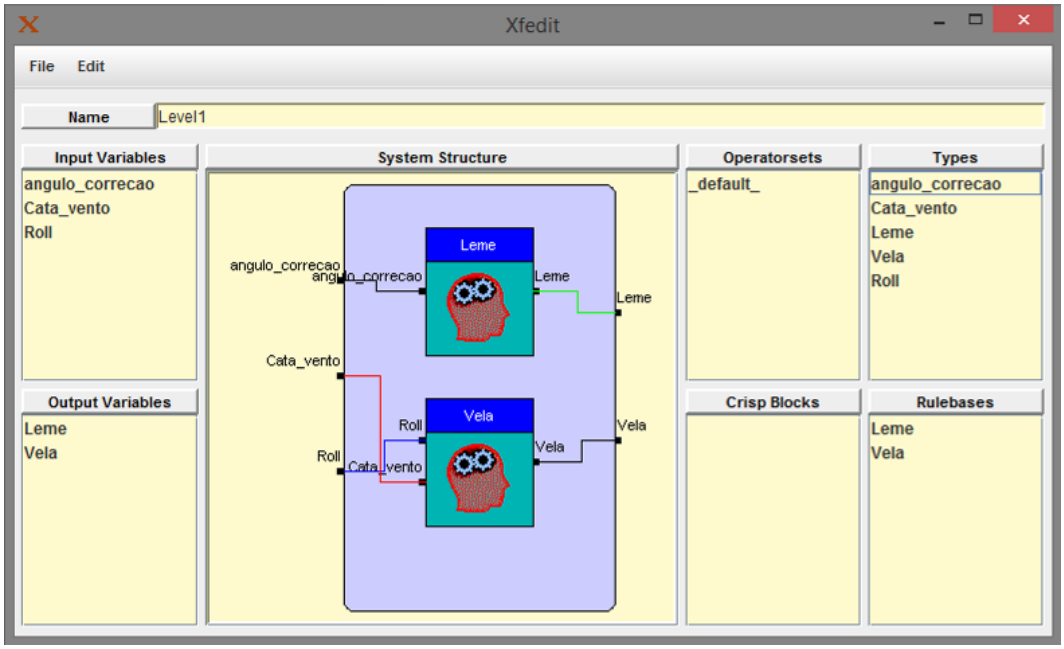


Figura 4.8: Janela de edição de sistemas XFuzzy [25].

4.2.2. Ferramenta de geração de código IOPT-Tools

A ferramenta IOPT-Tools [30] é uma ferramenta desenvolvida pelo grupo de investigação de sistemas reconfiguráveis e embutidos GRES [51] da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa.

Esta ferramenta consiste num IDE hospedado num servidor remoto que oferece um conjunto de utilidades para a criação, edição (figura 4.9) e análise de redes RdP, disponibilizando um simulador para análise comportamental, um gerador de espaços-estado para análise de árvore de resultados e um sistema de "queries" para análise condicional de sistemas de grande dimensão [52].

Além dos componentes descritos, esta ferramenta também permite a geração de código em linguagens C, VHDL e javascript correspondentes aos esquemas produzidos para fins de integração das redes desenvolvidas em sistemas de controlo externos.

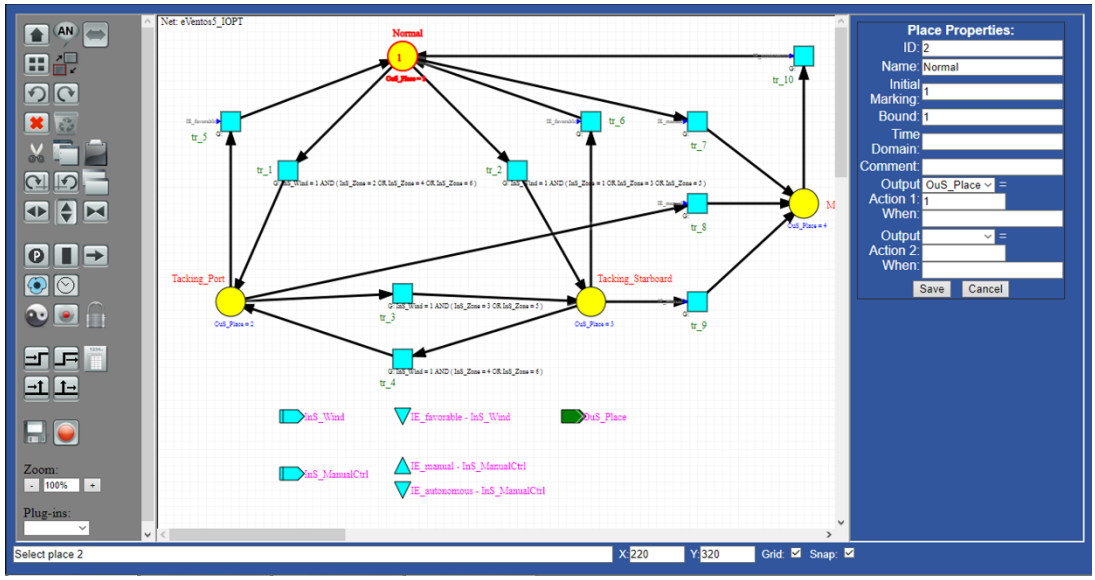


Figura 4.9: Editor da ferramenta IOPT-Tools [30].

Capítulo 5

Sistema de controlo

Este capítulo faz uma descrição do sistema de controlo desenvolvido nesta dissertação, mostrando a sua arquitetura geral, quer do ponto de vista dos componentes de hardware que o compõem, quer do código produzido, assim como as características da estratégia de controlo considerada.

Para a sua realização teve-se como ponto de partida uma análise do trabalho desenvolvido pelas dissertações passadas da linha eVentos a fim de se aproveitar ou reformular ideias com o intuito de conseguir um sistema de controlo funcional capaz de permitir que um veleiro navegue de forma autónoma ao longo de um trajeto pré-definido, tanto em condições de vento favorável como em condições de navegação à bolina, garantindo também o registo e confirmação de resultados dos valores do controlador.

5.1. Arquitetura do sistema

O sistema físico de controlo consiste numa plataforma Arduino MEGA 2560 detentor do código do controlador. A ela ligam-se três sensores, sendo estes um cata-vento, uma bússola magnética e um GPS assim como dois atuadores correspondentes aos servos do leme e da vela [53].

Além dos componentes descritos, o sistema conta também com outros dois subsistemas periféricos: o sistema de controlo manual remoto e uma aplicação de monitorização e registo de dados, de acordo com a figura 5.1.

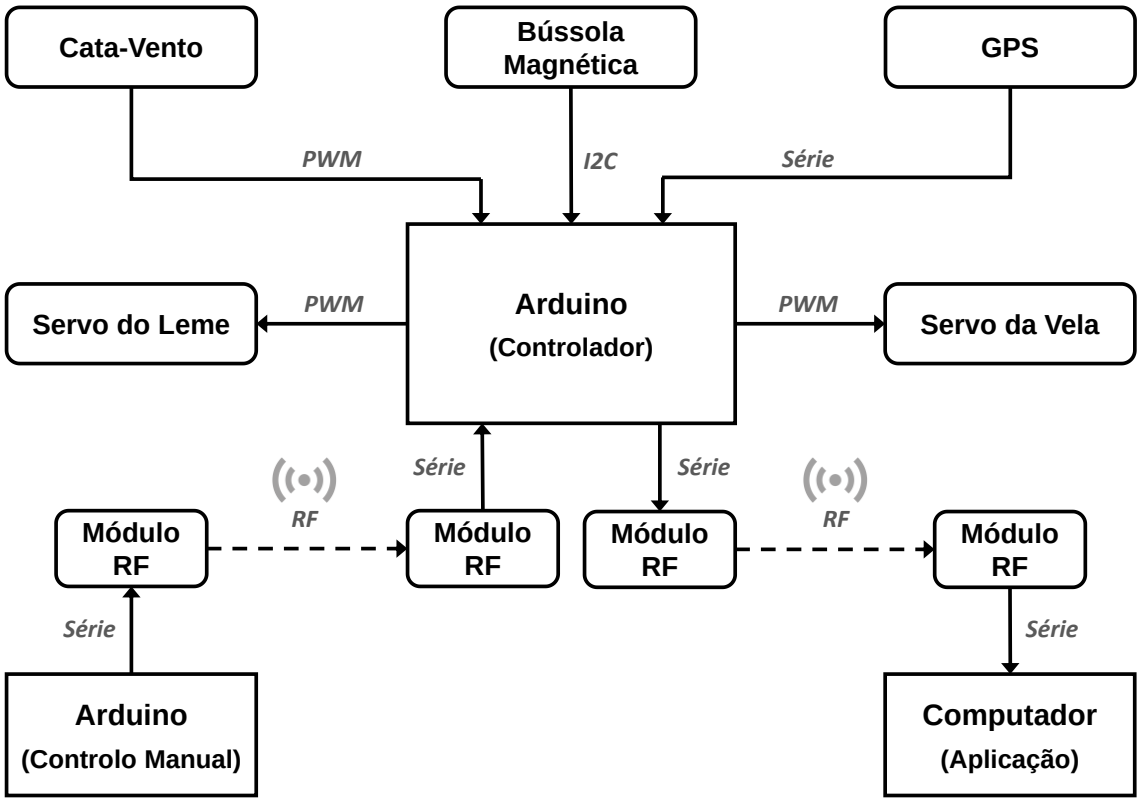


Figura 5.1: Diagrama de hardware do sistema de controlo.

O sistema de controlo manual é constituído por um Arduino de modelo semelhante ao do controlador sendo este detentor de um "joystick shield". Este é responsável pela comutação entre os modos de navegação autónoma e manual assim como do envio de comandos de controlo direto por parte do utilizador. A comunicação com o Arduino do controlador é feita por intermédio de um par de módulos de comunicação RF APC220 (um em cada Arduino), garantindo assim um maior alcance em comparação com Wi-Fi.

A aplicação de monitorização é albergada num computador, comunicando com o Arduino do controlador através de um par de módulos rádio semelhantes.

5.1.1. Estratégia de controlo

A estratégia de controlo é composta por um total de 6 elementos: três níveis principais resultantes da partição das ações de controlo autónomo do veleiro, dois elementos de natureza periférica e uma máquina de estados responsável pela coordenação dos níveis 1 e 2 de controlo assim como da comutação entre os modos de controlo autónomo e manual.

Tem-se assim os seguintes elementos:

- **Nível 1:** nível referente ao controlo difuso do leme e da vela que tem como objetivo levar o veleiro até à próxima coordenada de destino em condições de vento favorável;
- **Nível 2:** nível responsável pela deteção da condição de bolina, o cálculo do corredor de navegação, a determinação da zona interna do corredor onde se encontra o veleiro e o cálculo de coordenadas intermédias a fim de permitir o veleiro navegar em condições de vento desfavorável;
- **Nível 3:** nível responsável pela definição das coordenadas das diferentes etapas do trajeto e pela deteção da chegada do veleiro a cada uma delas;
- **Controlo Manual:** componente responsável pela comutação entre o modo de controlo manual, a fim de permitir o utilizador controlar o veleiro diretamente, e o modo de navegação autónoma;
- **Aplicação de Monitorização:** elemento responsável pela monitorização em tempo real e registo de resultados dos valores provenientes do controlador;
- **Máquina de estados:** RdP de quatro estados/lugares responsável pela coordenação entre o estado de navegação normal, os estados de navegação à bolina e o estado de controlo manual.

Estes elementos, juntamente com os seus relacionamentos, encontram-se representados na figura 5.2.

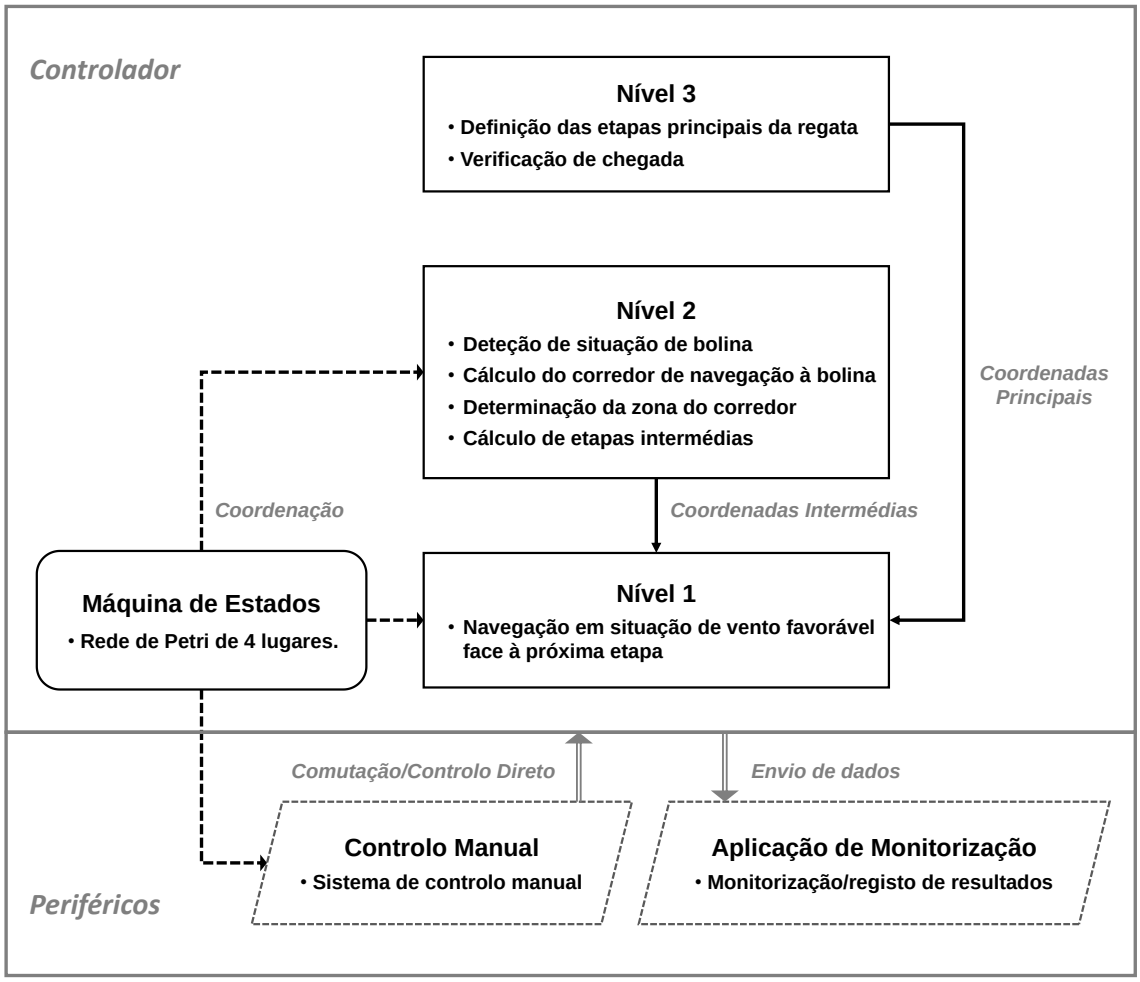


Figura 5.2: Estratégia de controlo.

5.1.2. Estrutura do código

O código C++ do controlador encontra-se organizado de forma modular com cada categoria de sensor e cada componente do sistema definido por uma classe distinta, em ficheiros separados, a fim de facilitar tanto a sua reutilização parcial como a adição de novos componentes. Além disto, conta também com alguns conjuntos de ficheiros "header-source" de linguagem C++ para a definição de bibliotecas auxiliares (GPS), o nível 1 de controlo (controlador XFuzzy) e para a integração do código referente à máquina de estados RdP da IOPT-Tools.

Todos os valores referentes à configuração das diferentes características do controlador, tais como limites de escalas de calibração, "baud rates", pinos dos diferentes sensores e outros valores intrínsecos à estratégia de navegação encontram-se num ficheiro "header" denominado *Consts.h*.

O ficheiro principal *main.ino* relaciona-se com cada classe por uma relação de associação (figura 5.3), utilizando uma instância de cada uma delas, tendo também uma relação de dependência para com o ficheiro *Consts.h* e os "headers" gerados pelo software XFuzzy e pela IOPT-Tools. O "header" da IOPT-Tools depende das classes do nível 2 e do joystick a fim de obter os seus sinais de entrada.

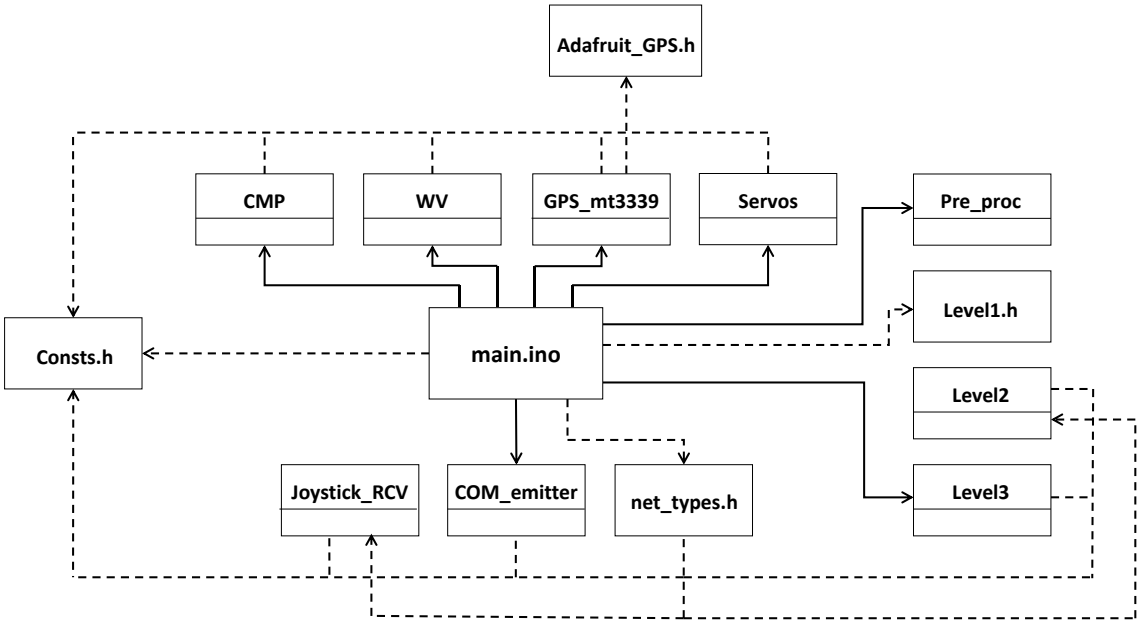


Figura 5.3: Diagrama conceptual de classes do controlador com alguns "headers" relevantes.

As classes *CMP*, *WV*, *GPS_mt3339* e *Servos* dizem respeito ao código sensorial da bússola, cata-vento, GPS e servos respetivamente.

O pré-processamento, responsável pelo cálculo do ângulo de correção, assim como os diferentes níveis do controlador encontram-se definidos em classes individuais de nome semelhante, sendo o nível 1 definido por um par de ficheiros "header-source" gerado pelo software XFuzzy.

Os códigos referentes à comunicação para com os componentes periféricos, responsáveis por receber informação do controlo manual e de enviar dados para a aplicação de monitorização, encontram-se definidos nas classes *Joystick_RCV* e *COM_emitter*.

A RdP e a biblioteca auxiliar do GPS estão declaradas pelos "headers" *net_types.h* (gerado pela aplicação IOPT-Tools) e *Adafruit_GPS.h* respetivamente.

Além do controlador, os dois sistemas periféricos também têm o seu código individual, cuja organização se encontra representada na figura 5.4. O sistema de controlo manual está presente no Arduino detentor do "joystick shield" e é composto por uma classe definidora dos métodos de envio de dados, um "main file" e, à semelhança do código do controlador, um "header" detentor de constantes relevantes.

A aplicação de monitorização é composta por duas classes: a classe *COM* que contem os métodos de processamento da trama recebida assim como de registo de dados e a classe *GUI* que tal como o nome indica é responsável pela apresentação e atualização do GUI da aplicação. Esta última usa uma instância da classe anterior.

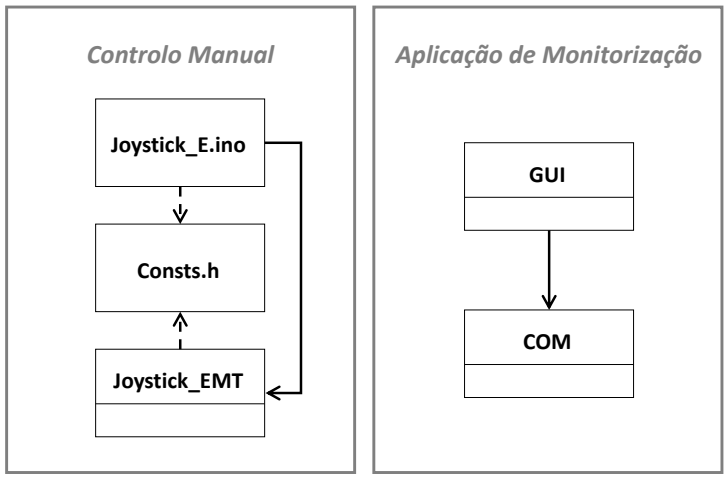


Figura 5.4: Diagrama conceptual de classes dos dispositivos periféricos.

5.2. Pré-processamento

O pré-processamento é o módulo responsável por achar a orientação do veleiro face ao destino, determinando o ângulo de correção necessário a fim de o controlador orientar o veleiro no sentido do mesmo. Este é calculado a partir do valor do "bearing" (orientação da proa face ao Norte magnético) fornecido pela bússola, as coordenadas geográficas da localização corrente do veleiro dadas pelo módulo GPS e a localização do próximo destino, devolvendo um ângulo cujo o valor se integra numa escala inserida no intervalo [-180, 180], à semelhança do que foi realizado em eVentos 3 [3].

O cálculo do pré-processamento é realizado pela função *preProc* da classe *Pre_proc* do código do controlador, sendo o ângulo de correção posteriormente utilizado como variável de entrada no controlador XFuzzy a fim de este determinar qual será a ação do leme.

5.2.1. Cálculo do ângulo do vetor de destino

O primeiro passo do módulo de pré-processamento é o cálculo do ângulo do vetor de destino. Este vetor tem origem no ponto geográfico onde se encontra o veleiro nesse instante e acaba no ponto do próximo destino que se pretende alcançar no decurso da navegação.

Para fins de código considerou-se que tanto o ângulo da proa como o ângulo do vetor de destino seguem a mesma escala angular de 0 a 360 graus com zero no sentido do Norte magnético [2], conforme apresentado na figura 5.5.

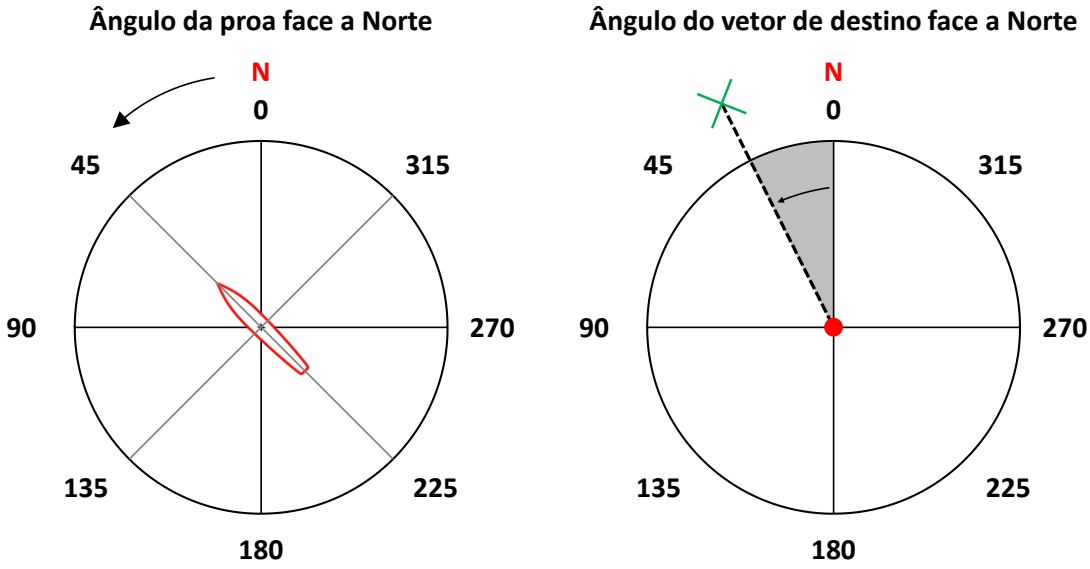


Figura 5.5: Referenciais angulares do ângulo da proa (bússola) *à esquerda* e do vetor de destino *à direita* face a Norte.

A função começa por calcular as diferenças de longitude (*dx*) e latitude (*dy*) entre o ponto de destino e o ponto atual do veleiro. De seguida verificará os casos particulares em que o vetor de destino coincide com a fronteira entre quadrantes (*dx* ou *dy* nulos) e, caso se encontre numa destas situações, considerará o devido ângulo (0º, 90º, 180º ou 270º).

Caso não se encontre em nenhuma destas situações, a função calculará o ângulo de destino a partir de uma projeção para o primeiro quadrante, fazendo *q=abs(dy/dx)* e por fim *atan(q)* [2], convertendo de seguida o ângulo em graus.

Após a determinação do ângulo, a função verificará pelos sinais das diferenças de coordenadas em que quadrante da escala alvo o vetor de destino realmente se encontra a fim de aplicar as compensações angulares indicadas na figura 5.6.

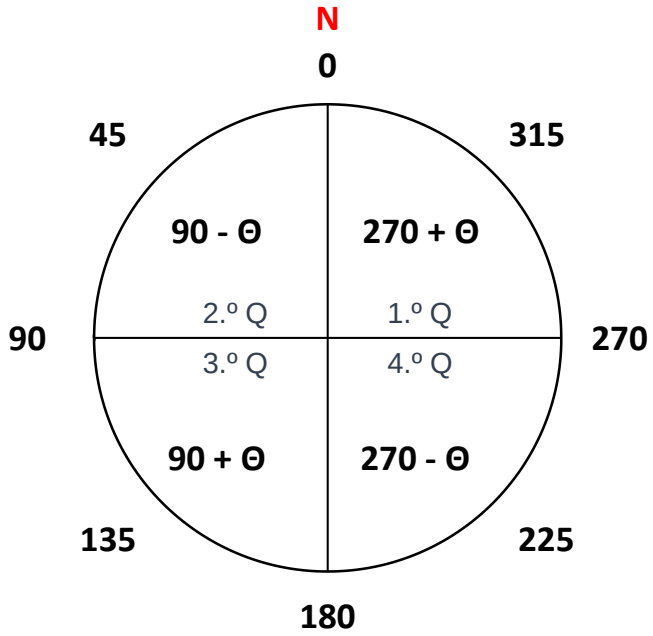


Figura 5.6: Compensações do ângulo do vetor de destino para diferentes quadrantes.

5.2.2. Cálculo do ângulo de correção

O último passo da função *preProc* será o cálculo do ângulo de correção a fim de este poder ser transmitido ao controlador do leme. O ângulo de correção será calculado a partir do ângulo da orientação da proa face ao Norte (bússola) e o ângulo de orientação do destino, tal como foi proposto por eVentos 2 e implementado em eVentos 3 [2], [3] de acordo com a fórmula 5.1.

$$\textbf{ângulo de correção} = \textbf{ângulo da bússola} - \textbf{ângulo do destino} \qquad (5.1)$$

À semelhança dos projetos anteriores, o resultado será expresso numa escala simétrica desde -180 a 180, cuja leitura indicará ou uma viragem a bombordo para números negativos, ou uma viragem a estibordo para números positivos [2], [3] de acordo com a figura 5.7.

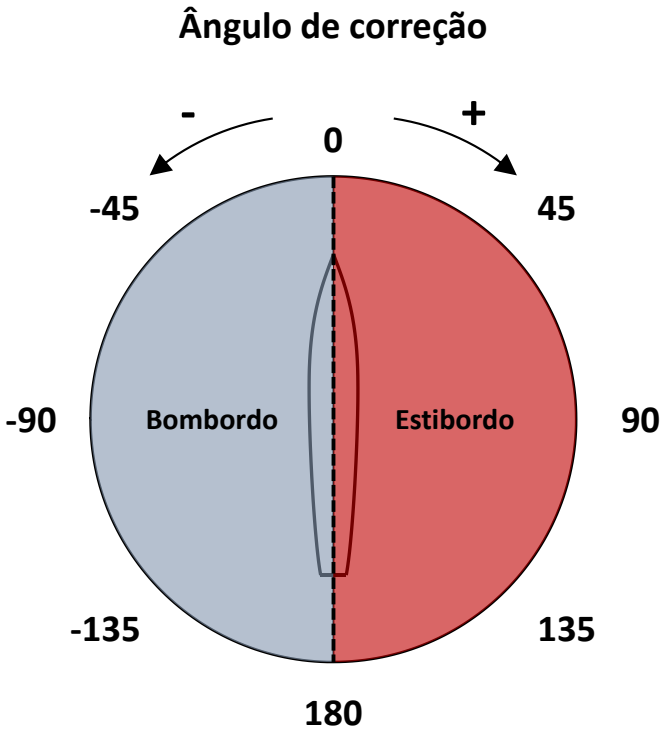


Figura 5.7: Escala angular do ângulo de correção [2], [3].

Uma vez feito este cálculo há que ter em conta dois casos particulares a fim de transpor corretamente o valor para a escala desejada. Por vezes o valor resultante da diferença resultará num valor fora do intervalo [-180, 180] pelo que será necessário achar o valor correto à escala alvo.

Conforme mostrado na figura 5.8, caso o valor seja maior que 180, quando o ângulo dado pela bússola é muito superior, subtrai-se 360. Caso o valor seja menor que 180, quando o ângulo face ao destino é muito superior, soma-se 360.

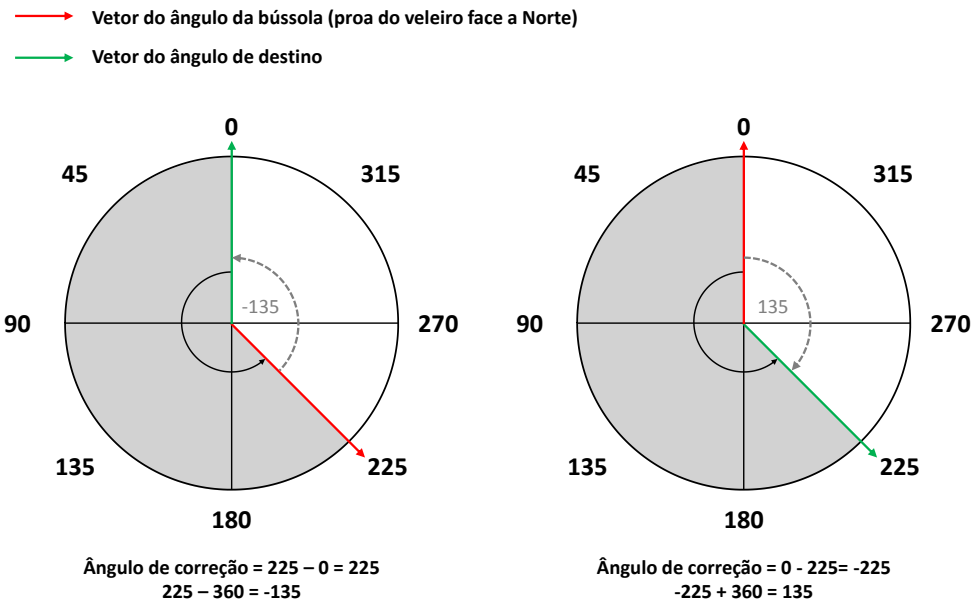


Figura 5.8: Exemplos particulares de escala excedida no cálculo do ângulo de correção.

5.3. Nível 1

O nível mais baixo do controlador, denominado nível 1, é o nível responsável pelo controlo autónomo em condições de vento favorável. Este tem como objetivo orientar o veleiro em função do destino a atingir e caçar ou folgar a vela mediante o estado do cata-vento e do "roll" lido pela bússola, consistindo num controlador difuso desenvolvido pela plataforma XFuzzy, sendo composto essencialmente por dois componentes representados na figura 5.9: o controlo do leme e o controlo da vela.

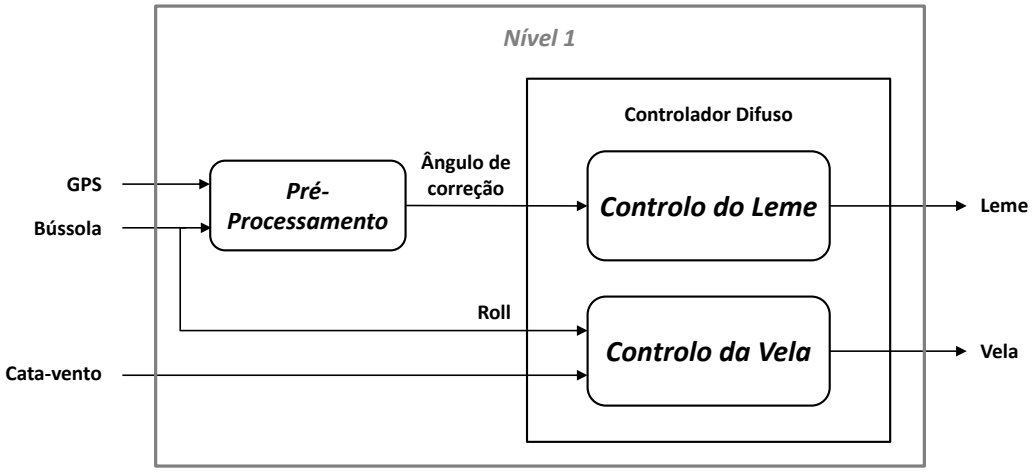


Figura 5.9: Sistema do controlador difuso do nível 1 desenvolvido com software XFuzzy.

O controlo do leme tem como variável de entrada o ângulo de correção a fim de saber a orientação do destino face à proa do veleiro. Este determina o valor de saída do leme a enviar ao servo responsável pela sua manipulação. Quanto mais à esquerda o destino se encontre, mais a bombordo o veleiro vira e quanto mais à direita esteja o destino mais a estibordo o veleiro virará.

O controlo da vela tem como variáveis de entrada a posição do cata-vento e o valor do "roll" a fim de determinar o valor de saída da vela. Quanto mais próximo da proa estiver a pá do cata-vento (vento aparente de ré) mais a vela se folgará a fim de aproveitar a impulsão do vento e quanto mais próxima da ré do veleiro esta estiver (vento aparente de proa) mais a vela se fechará a fim de evitar o deslocamento no sentido contrário do pretendido. O "roll" por sua vez determinará se a vela segue o seu comportamento normal para inclinações aceitáveis ou, em caso de inclinações excessivas e independentemente do estado do cata-vento, se esta se folgará a fim de evitar a perda de controlo do veleiro.

As características das funções pertença usadas neste controlador são da autoria da dissertação eVentos 3 [3] sendo que a dissertação corrente difere desta última em dois pontos: o considerar da posição do cata-vento como variável de entrada do controlo da vela (em vez da posição angular de incidência do vento aparente) e também da junção da variável de controlo "roll" que, apesar de não implementada em eVentos 3, foi dada como proposta para projetos futuros.

5.3.1. Controlo do leme

O controlo do leme vai receber como variável de entrada o valor do ângulo de correção calculado pelo módulo de pré-processamento e, de acordo com as regras de inferência, determina o respetivo sinal PWM a enviar ao servo do leme.

5.3.1.1. Funções pertença da variável Ângulo de correção

Sabendo que o ângulo de correção vai tomar um valor numa escala situada no intervalo [-180, 180], utilizaram-se 7 funções pertença de forma a abranger diferentes intervalos da mesma [3], conforme ilustrado na figura 5.10.

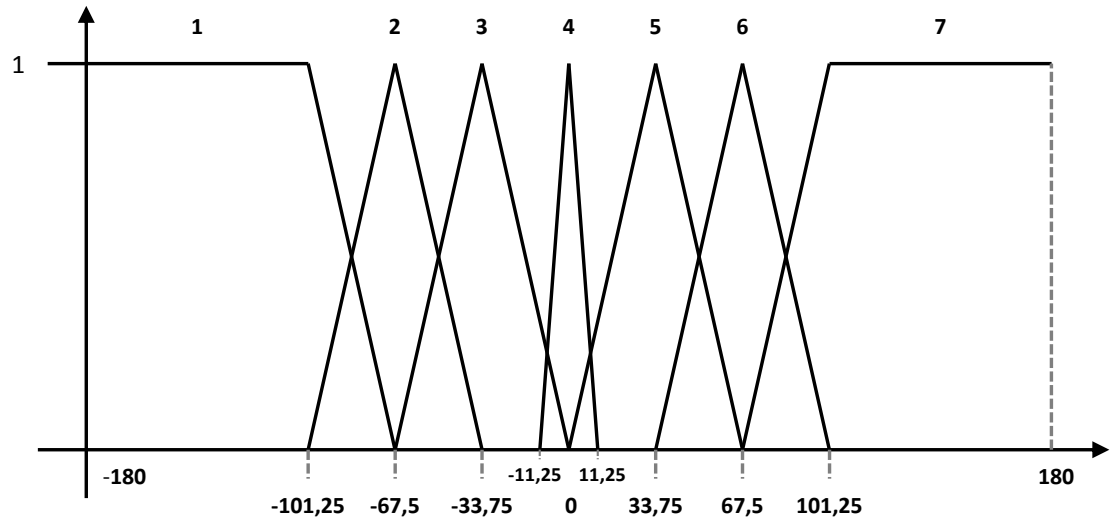


Figura 5.10: Funções pertença do ângulo de correção segundo eVentos 3 [3].

De acordo com a tabela 5.1, é associado a cada função pertença um termo linguístico referente ao sentido da viragem.

Tabela 5.1: Termos linguísticos do ângulo de correção [3].

Numeração	Termo Linguístico
1	Tudo à esquerda
2	Esquerda
3	Um pouco à esquerda
4	Centrado
5	Um pouco à direita
6	Direita
7	Tudo à direita

Estes termos vão definir os antecedentes a associar com os consequentes referentes ao atuador do leme.

5.3.1.2. Funções pertença da variável Leme

De forma a associar cada função de entrada do ângulo de correção com uma função de saída, por intermédio de regras de inferência, definiram-se 5 funções pertença para o leme [3] (figura 5.11), cada qual com o seu termo linguístico (tabela 5.2). Este, juntamente com a vela, tem uma escala de valores no intervalo [1000, 2000]. Neste caso, o limite inferior corresponde a uma viragem total a bombordo e o limite máximo corresponde a uma viragem total a estibordo.

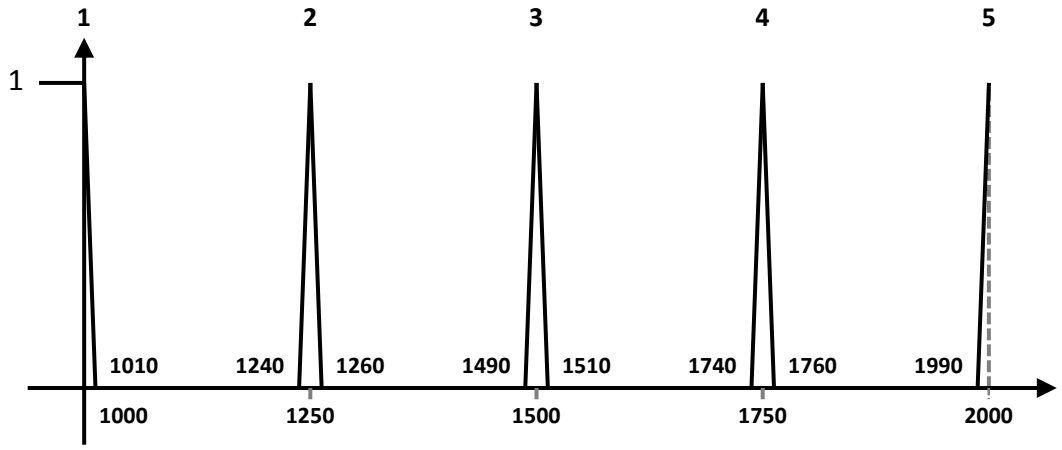


Figura 5.11: Funções pertença do leme baseadas em eVentos 3 [3].

Tabela 5.2: Termos linguísticos do leme [3].

Numeração	Termo Linguístico
1	Tudo a bombordo
2	Bombordo
3	Centrado
4	Estibordo
5	Tudo a estibordo

5.3.1.3. Regras de inferência do controlo do leme

As regras de inferência do controlo do leme serão responsáveis pela definição dos consequentes a tomar (funções pertença do leme) de acordo com os antecedentes por ele recebidos (funções pertença do ângulo de correção).

Definiram-se assim um total de 7 regras de inferência, estando estas indicadas na tabela 5.3.

Tabela 5.3: Regras de inferência do controlo do leme [3].

Ângulo de correção (*antecedente*)	**Leme (*consequente*)**
Tudo à esquerda	Tudo a bombordo
Esquerda	Tudo a bombordo
Um pouco à esquerda	Bombordo
Centrado	Centrado
Um pouco à direita	Estibordo
Direita	Tudo a estibordo
Tudo à direita	Tudo a estibordo

De notar que cada par de antecedentes correspondentes às duas funções de cada extremo vão ser associadas com o mesmo consequente, à semelhança do que foi definido em eVentos 3 [3].

5.3.2. Controlo da vela

O controlo da vela vai receber como entrada as variáveis referentes ao valor do cata-vento e o valor do "roll".

O valor do cata-vento dado pelo magnetómetro situa-se numa escala simétrica de 0º (cata-vento à proa) a 180º (cata-vento à ré). O valor do "roll" é fornecido pela bússola e situa-se numa escala de 0º (direito) a 90º (tombado), conforme apresentado na figura 5.12.

De acordo com estes valores, o controlador irá modificar o valor PWM a enviar para o servo que controla a vela, dando sempre prioridade ao folgar da vela caso o "roll" seja muito elevado, independentemente do valor do cata-vento.

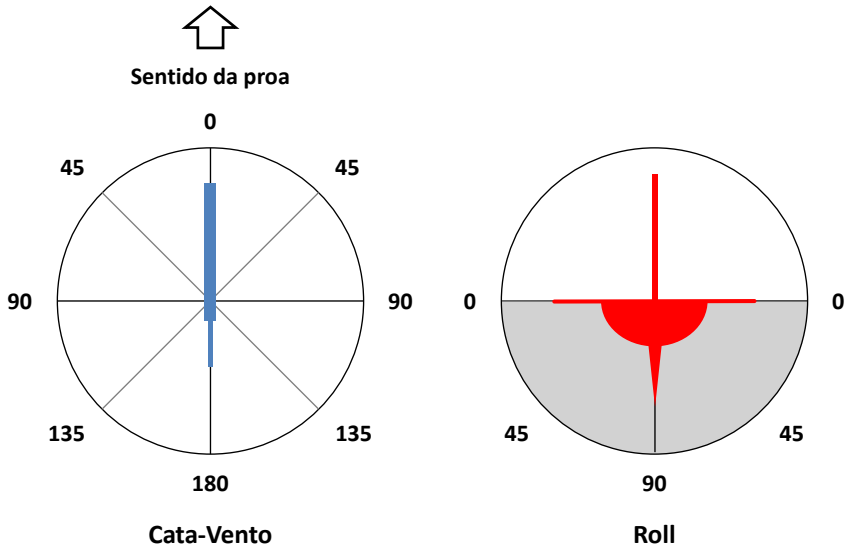


Figura 5.12: Escala angular do cata-vento *à esquerda* e escala angular do "roll" *à direita* [3].

5.3.2.1. Funções pertença da variável Cata-vento

A variável *cata-vento* é definida por três funções pertença que abrangem diferentes intervalos angulares correspondentes à posição da pá do sensor de mesmo nome em função do veleiro (figura 5.13). A cada uma destas é associada um termo linguístico distinto de acordo com a tabela 5.4.

Estas funções pertença são baseadas na definição das funções pertença da variável *Ângulo de orientação do vento* da estratégia de controlo da dissertação eVentos 3 [3], apesar desta última ser referente ao vento aparente.

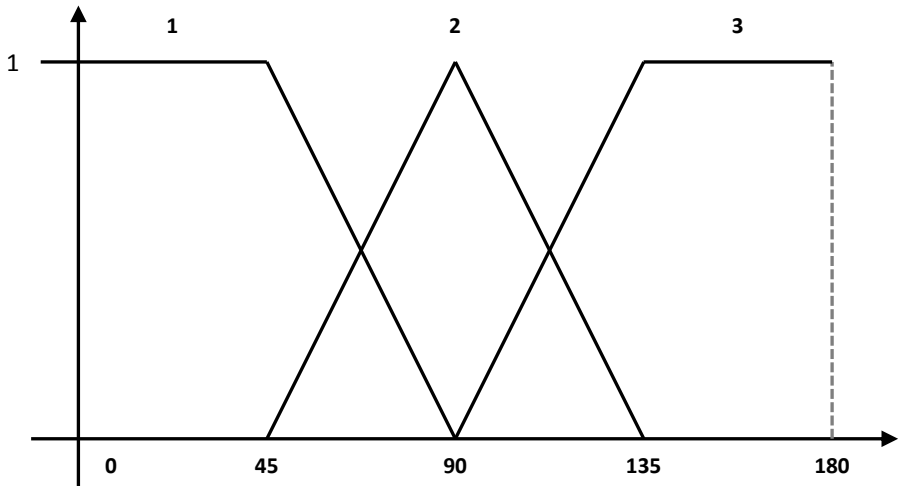


Figura 5.13: Funções pertença do cata-vento baseadas na variável *Ângulo de orientação do vento* de eVentos 3 [3].

Tabela 5.4: Termos linguísticos do cata-vento.

Numeração	Termo Linguístico
1	Cata-vento à Proa
2	Cata-vento de Través
3	Cata-vento à Ré

5.3.2.2. Funções pertença da variável "Roll"

O "roll" é dado por uma escala simétrica no intervalo [0, 90], marcando zero quando a bússola se encontra perfeitamente na horizontal, sendo definido pelas duas funções pertença apresentadas na figura 5.14.

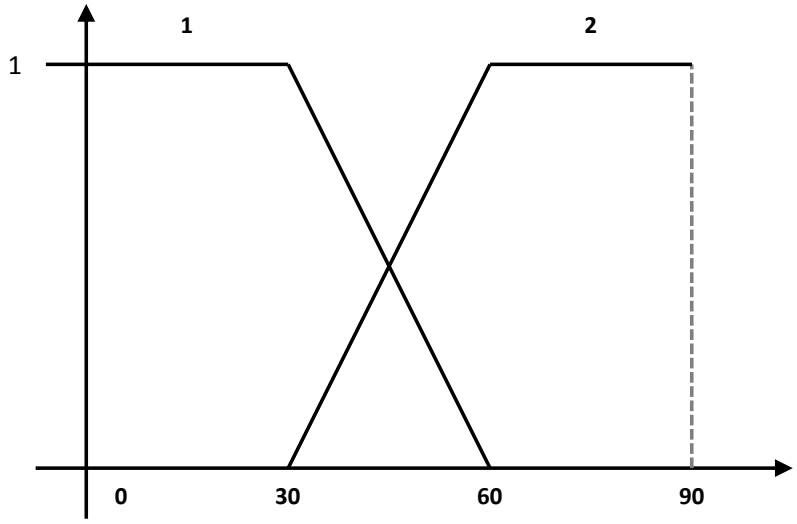


Figura 5.14: Funções pertença do "roll" sugeridas por eVentos 3 [3].

Caso o "roll" iguale ou exceda 30º considera-se que o veleiro está a entrar na zona crítica de inclinação pelo que se pretende que a vela folgue a fim de impedir o vento de contribuir para inclinações excessivas, tendo-se assim dois termos linguísticos de acordo com a tabela 5.5.

Tabela 5.5: Termos linguísticos do "roll" [3].

Numeração	Termo Linguístico
1	Inclinação Aceitável
2	Inclinação Crítica

5.3.2.3. Funções pertença da variável Vela

A variável *vela* diz respeito ao intervalo de valores entre 1000 e 2000 do servo responsável por caçar/folgar as velas do veleiro. Para esta utilizam-se três funções pertença (figura 5.15) a fim de associar com as funções das variáveis *cata-vento* e *"roll"*.

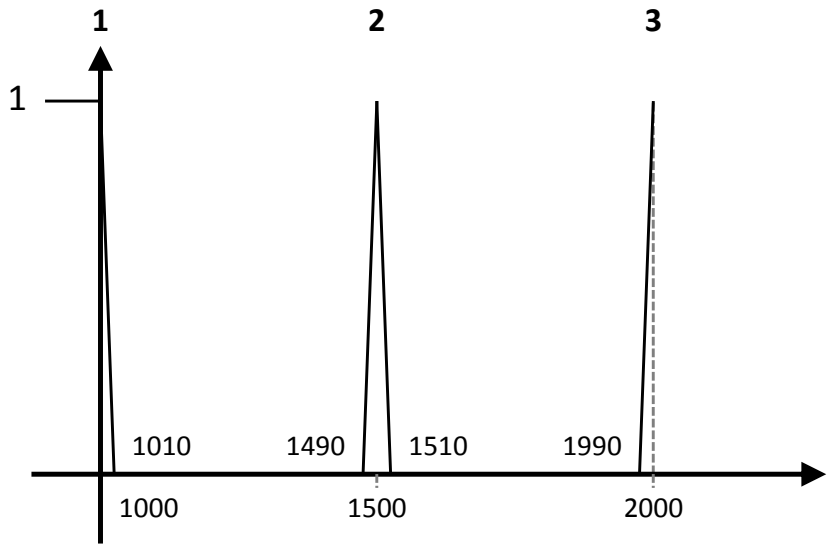


Figura 5.15: Funções pertença da vela [3].

A cada uma delas associa-se um termo linguístico referente à posição da vela, conforme descrito na tabela 5.6.

Tabela 5.6: Termos linguísticos da vela [3].

Numeração	Termo Linguístico
1	Vela Caçada
2	Vela Centrada
3	Vela Folgada

5.3.2.4. Regras de inferência do controlo da vela

Relacionando as funções pertença do cata-vento e do "roll", utilizando operadores lógicos "AND" para relacionamento de antecedentes, definiu-se um total de 3×2 regras de inferência (tabela 5.7), definindo todas as situações possíveis do estado da vela consoante os valores de ambas as variáveis de entrada.

Tabela 5.7: Regras de inferência do controlo da vela.

Antecedentes (&&)		Consequentes
Vento	**Roll**	**Vela**
Cata-vento à Proa	Inclinação Aceitável	Vela Folgada
Cata-vento a Través	Inclinação Aceitável	Vela Centrada
Cata-vento à Ré	Inclinação Aceitável	Vela Caçada
Cata-vento à Proa	Inclinação Crítica	Vela Folgada
Cata-vento a Través	Inclinação Crítica	Vela Folgada
Cata-vento à Ré	Inclinação Crítica	Vela Folgada

5.4. Nível 2

O nível 2 do controlador é o nível responsável por permitir o veleiro ser capaz de navegar à bolina em condições de vento desfavorável, i.e., dentro da zona proibida. Este módulo é definido pela classe *Level2* do código de controlo e é responsável por 4 operações: determinação da condição de bolina, cálculo de um corredor de navegação, determinação da zona do corredor onde o veleiro se encontra e cálculo de coordenadas intermédias de bolina.

O corredor de navegação definido por este nível parte da nomenclatura base definida em eVentos 4 [4], tendo sido feito uma alteração dos métodos de cálculo e de algumas características do corredor a fim de garantir maior flexibilidade, precisão de cálculo e robustez face a certos casos particulares de navegação.

A determinação da condição de bolina é feita em duas fases. Numa primeira fase é verificado se o cata-vento se encontra dentro da zona proibida de navegação em função do ponto de destino a atingir e, caso esta se verifique, é então calculado o corredor de navegação. Numa segunda fase, após o cálculo do corredor, é verificado se a condição de bolina se mantém em função do sentido do próprio corredor.

O corredor de navegação é constituído por 6 retas: uma reta principal, duas retas de limite do corredor, uma reta de meta e duas retas de estreitamento. Este tem como função manter o veleiro a navegar dentro das retas de limite e estreitamento até à próxima etapa a atingir [4].

Sendo que a navegação à bolina ocorre numa trajetória em ziguezague, é necessário o cálculo de coordenadas intermédias de forma a permitir o veleiro navegar contra o vento ao longo do corredor. Estas, juntamente com a divisão do corredor em zonas internas distintas, vão permitir ao veleiro atingir a etapa desejada em condições de vento desfavorável.

5.4.1. Deteção da condição de bolina

Para o controlador saber se o veleiro se encontra a navegar contra o vento, a fim de efetuar o cálculo do corredor de navegação, é necessário um método que permita a este saber se o veleiro se encontra dentro da zona proibida.

Numa primeira fase, quando o vento é favorável à navegação antes da mudança de estado, interessa saber qual a posição do cata-vento em função do valor do ângulo de correção mediante a margem angular da zona proibida. Numa segunda fase, caso o corredor já tenha sido calculado, interessa saber qual a posição do cata-vento em função do sentido do corredor, tendo em conta também a mesma margem.

Este cálculo é realizado pelo método *intoWind*, tendo como parâmetros o ângulo de correção, o ângulo da bússola (neste caso numa escala [-180, 180]), o valor do cata-vento em escala [0, 360] e um booleano de controlo *tacking_mode* para identificação da fase. De notar que o ângulo de correção tem o zero da escala à proa do veleiro e o ângulo da bússola tem o zero a Norte.

5.4.1.1. Fase 1: estado de bolina em função do ponto de destino

O primeiro passo a realizar a fim de saber se o veleiro se encontra a navegar à bolina é a verificação do valor do ângulo de correção atual.

De seguida, subtraindo esse valor a 180, acha-se a projeção desse valor na escala do cata-vento. A partir do valor da projeção calculam-se as margens mínima e máxima correspondentes à zona proibida de navegação, subtraindo e somando respetivamente o ângulo de bolina pretendido. Caso estas passem o limite da escala, quer por excesso ou por defeito, subtrai-se ou soma-se 360 ao valor respetivamente.

No fim é verificado se o valor do cata-vento fornecido pelo sensor se encontra dentro do intervalo definido pelas margens (ex. da figura 5.16) e, caso se verifique, é alterado o booleano de controlo para o valor "true" a fim de sinalizar o estado de bolina. Nesta fase são também guardados os valores da bússola e das margens da zona proibida para utilização na segunda fase.

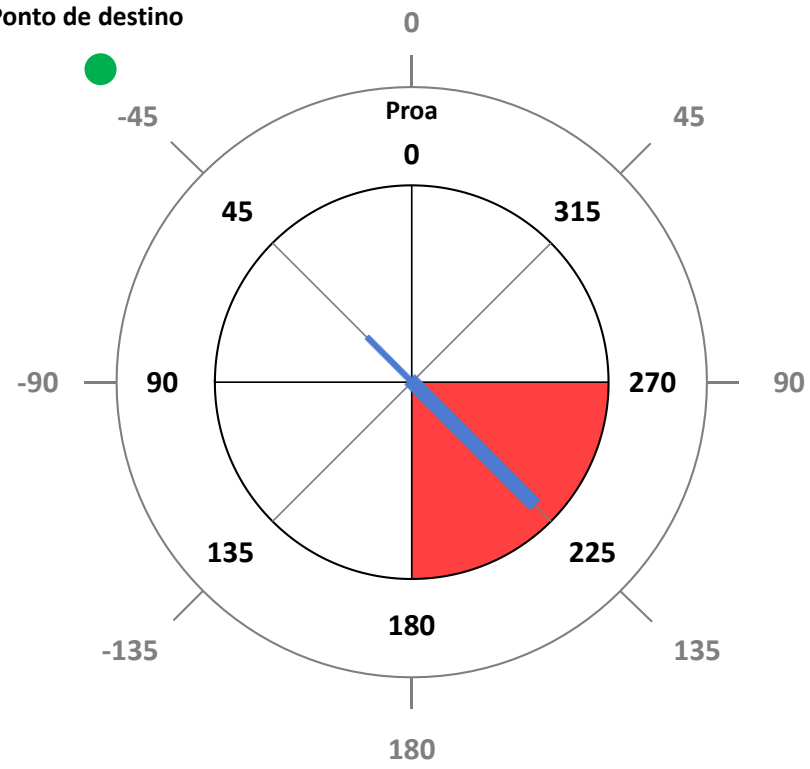


Figura 5.16: Exemplo de situação de bolina com ângulo de correção de -45º (*escala exterior*) e zona proibida de 45º (*escala interior do cata-vento*).

5.4.1.2. Fase 2: estado de bolina em função do corredor

Assim que o veleiro se encontre em situação de navegação à bolina, pretende-se saber qual o valor do cata-vento em função do corredor.

Quando o veleiro começa a sua trajetória em ziguezague ao longo do corredor calculado, o valor do ângulo de correção irá sofrer alterações à medida que o veleiro muda a sua posição relativa ao ponto de destino, pelo que este não será o ângulo indicado a utilizar como referência.

A solução a este problema parte por utilizar os valores da bússola e das margens guardadas na primeira fase. Utilizando como referência o valor da bússola anteriormente guardado, verifica-se que quando este aumenta de valor as margens da zona proibida sofrerão um aumento na mesma medida em função dos valores da primeira fase e quando o valor da bússola diminui estas irão também diminuir, conforme mostrado no exemplo da figura 5.17.

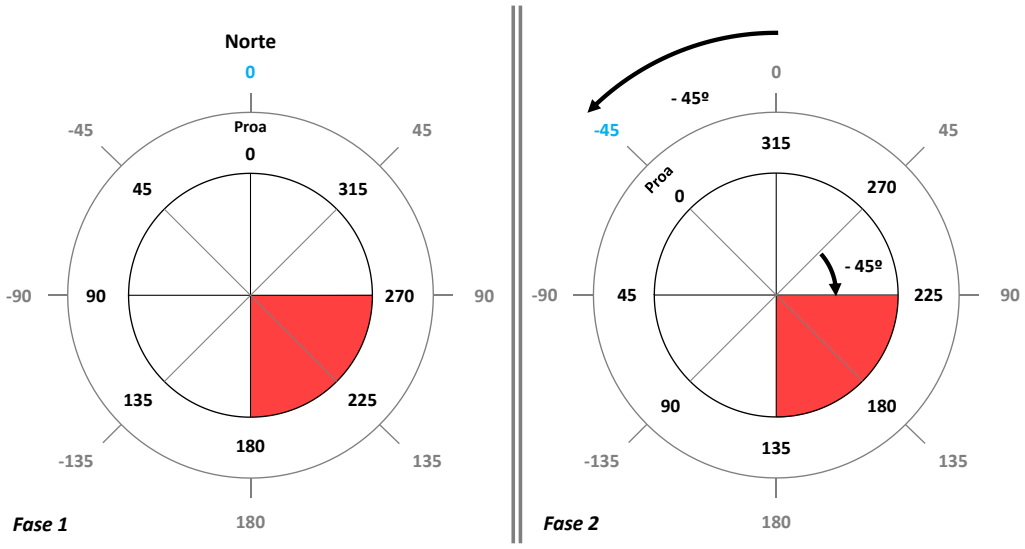


Figura 5.17: Exemplo de compensação de margens por defeito da primeira para a segunda fase.

Assim acha-se a compensação a dar ao valor da bússola pela diferença entre o seu valor atual e o valor da mesma guardado na primeira fase. Caso a diferença exceda os 180º subtrai-se 360 e se for menor que -180º soma-se 360.

De seguida determinam-se as margens da zona proibida, subtraindo ou somando o valor anterior às margens guardadas na primeira fase. Caso estas excedam o limite de escala de 360º são-lhes subtraídas esse mesmo valor e caso sejam negativas soma-se 360.

No fim é verificado se o valor do cata-vento se encontra dentro das novas margens calculadas. Caso não se encontre à bolina em função do corredor, o controlador volta então à fase 1.

5.4.2. Cálculo do corredor

Dada a necessidade de restringir a trajetória à bolina do veleiro, de modo a que esta ocorra dentro de uma área retangular pré-definida, definiu-se um corredor de navegação. Este corredor parte da nomenclatura base presente da dissertação eVentos 4 [4] e resulta de uma reformulação dos métodos de cálculo utilizados a fim de garantir maior versatilidade e robustez.

Analisando a solução proposta por eVentos 4 para o cálculo do corredor verificaram-se dois problemas a ter em conta para a estratégia de controlo. Em primeiro lugar o corredor não permite assimetria, quer dos ângulos das retas de estreitamento, quer da distância das balizas ao ponto de destino, quer dos limites laterais do corredor em função da reta principal. A capacidade de calcular um corredor assimétrico poderá ser útil para situações de navegação futuras onde seja necessário compensar o efeito das correntes marítimas através do aumento das margens no sentido da corrente, garantindo assim mais versatilidade ao controlador.

Em segundo lugar o cálculo do corredor não tem em conta situações em que pelo menos uma das suas retas constituintes possa ser vertical. Sendo que cada reta é definida pela sua equação reduzida (variáveis de *declive* e *ordenada na origem*), tem-se nestas situações um declive não definido pelo que será necessário achar uma solução para o controlador conseguir tratar estes casos particulares.

Para lidar com o problema de verticalidade das retas consideraram-se três soluções possíveis, analisando-se as diferentes vantagens e desvantagens de cada uma a fim de escolher o método a utilizar.

Têm-se assim as seguintes possibilidades:

- **Ignorar o ciclo corrente:** caso a situação de reta vertical se verifique, ignora-se o ciclo atual, esperando que o próximo obtenha valores diferentes por parte dos sensores. Apesar de ser o método mais simples e de maior rapidez de cálculo, ocorre o risco de o controlador ficar parado caso a leitura sensorial não permita a aquisição de valores diferentes;

- **Soma de pequena incerteza:** ao somar uma pequena incerteza ao valor do declive consegue-se um valor aproximado da reta, não resultando em paragem do controlador, contudo a precisão do cálculo será afetada por este método já que o erro da incerteza é somado a erros provenientes de arredondamentos de cálculo, além de ser dependente das características da arquitetura do microcontrolador em função do tipo de variável usada;

- **Estratégia robusta:** método implementado no controlador desenvolvido, verificando os casos particulares em que esta situação ocorre e guardando o valor longitudinal das respetivas retas verticais. Apesar de ser mais complexo, este permite também uma maior precisão de cálculo, quer para contextos de navegação como para testes laboratoriais.

O corredor é assim constituído por um total de 6 retas e duas balizas de aproximação, conforme representado na figura 5.18. Estas retas têm como função definir áreas internas do corredor a fim permitir ao controlador desempenhar a sua trajetória à bolina.

A reta principal r_0 é a reta que une o ponto de localização do veleiro no instante do cálculo do corredor (ponto de origem do corredor) ao ponto da próxima etapa a atingir (ponto de destino). A partir desta definem-se duas retas de limite do corredor (r_1 e r_2) situadas a uma distância d_1 e d_2 respetivamente da reta principal, sendo estas três retas paralelas umas às outras.

Ortogonalmente às retas anteriores e passando no ponto de destino define-se uma reta de meta r_3.

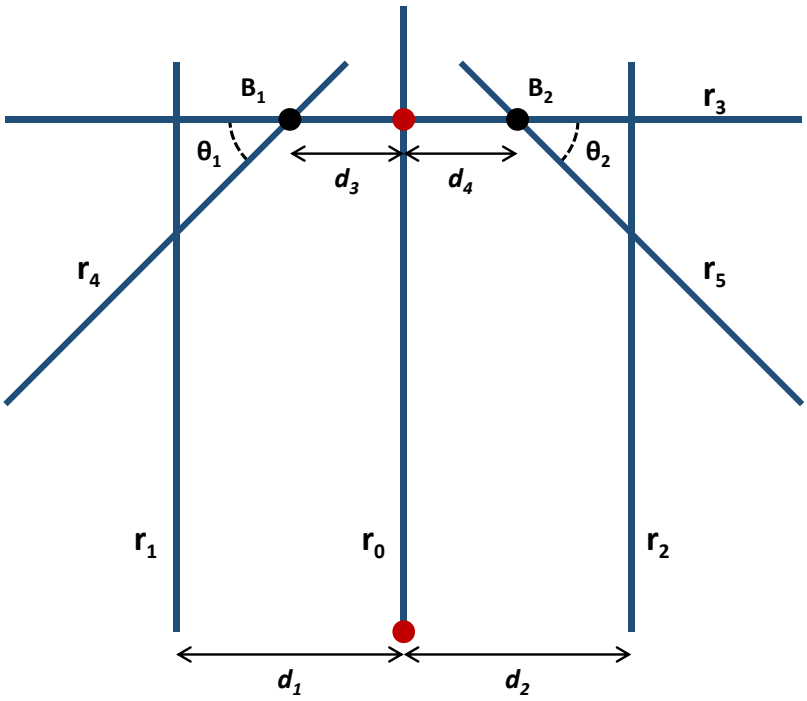


Figura 5.18: Corredor de navegação à bolina e seus constituintes.

Ao longo da reta de meta existem duas balizas de aproximação laterais: a baliza B_1 situada entre as retas r_0 e r_1, posicionada a uma distância d_3 do ponto de destino, e a baliza B_2 situada entre as retas r_0 e r_2, posicionada a uma distância d_4 do mesmo ponto. Passando por estas balizas têm-se as retas de estreitamento r_4 e r_5 a fim de fazer o veleiro convergir para o ponto de destino, sendo estas resultantes de uma rotação angular feita a partir da reta de meta.

O cálculo do corredor é efetuado pelo método *calculateCorr* que recebe as coordenadas dos pontos de origem (GPS) e destino (Nível 3) além de fazer uso de constantes definidoras das características do corredor pretendido.

5.4.2.1. Cálculo das retas de limite, meta e balizas de aproximação

Dada a necessidade de verificação dos casos particulares em que pelo menos uma reta possa ser vertical definiram-se 8 situações distintas que o corredor pode tomar, mediante a posição do ponto de destino em função do ponto de origem, num referencial angular considerado a partir dos eixos de latitude e longitude. Estas situações estão descritas na figura 5.19.

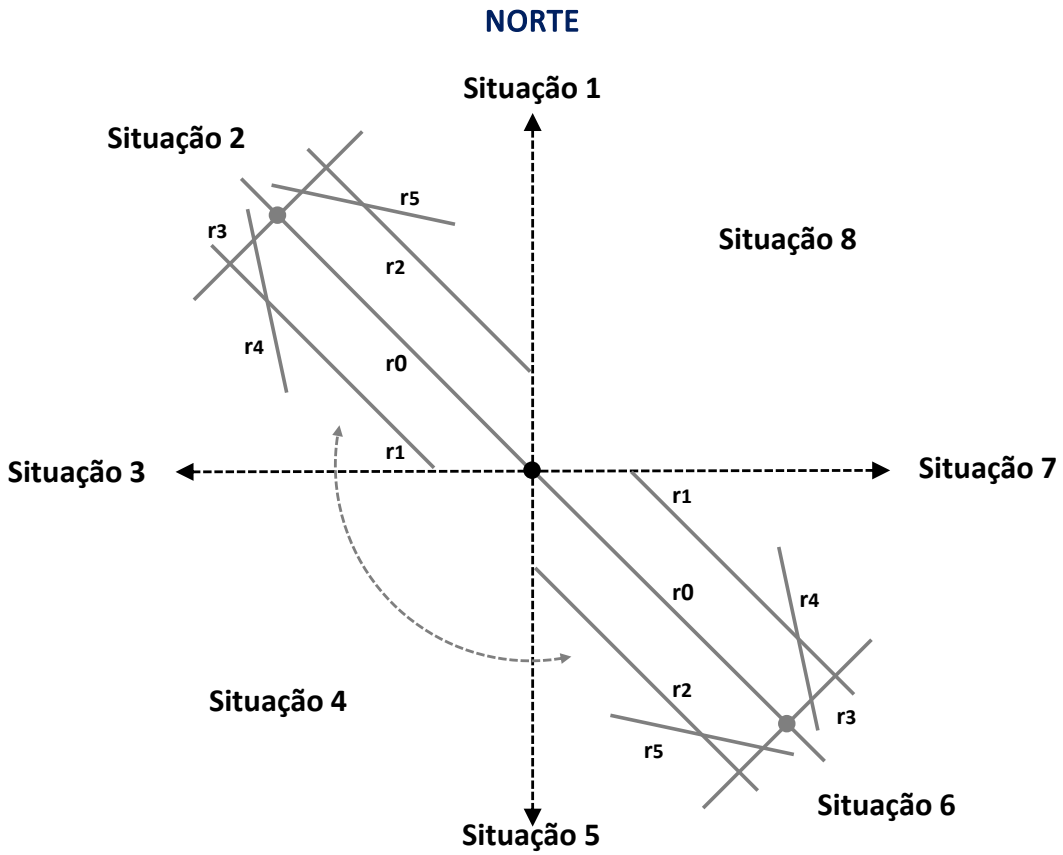


Figura 5.19: Situações possíveis do corredor no referencial angular latitude-longitude.

A função *calculateCorr* começa por verificar se o corredor se encontra nas situações 1, 3, 5 ou 7. Caso o corredor se encontre numa destas situações será calculado de imediato as retas r_0, r_1 r_2 e r_3 juntamente com as coordenadas das balizas B_1 e B_2.

Se o corredor se encontrar numa das outras situações descritas (reta principal diagonal) determinam-se as diferenças latitudinal (*dy*) e longitudinal (*dx*) entre os pontos de destino e origem do corredor a fim de determinar o declive das retas r_0, r_1 e r_2, assim como a ordenada na origem da reta principal, segundo a fórmula 5.2.

O declive da reta de meta r_3 é determinado a partir da sua relação de ortogonalidade para com o declive da reta r_0, de acordo com a fórmula 5.3.

$$\begin{gathered}\mathbf{m_0 = dy/dx} \\ \mathbf{b_0 = latitude\ de\ origem - m_0 \times longitude\ de\ origem}\end{gathered} \tag{5.2}$$

$$\begin{gathered}\mathbf{m_3 = -1/m_0} \\ \mathbf{b_3 = latitude\ de\ destino - m_3 \times longitude\ de\ destino}\end{gathered} \tag{5.3}$$

De seguida é necessário fazer o cálculo das retas de limite r_1 e r_2 a par com duas retas auxiliares r_1' e r_2' que irão cruzar com a reta de meta a fim de determinar as balizas de aproximação. Partindo da equação 5.4 da distância de uma reta a um ponto [54], representada pela sua equação completa, deduz-se a equação 5.5 da distância entre duas retas paralelas, em função das respetivas equações reduzidas.

$$\mathbf{d} = \frac{|\mathbf{Ax_1 + By_1 + C}|}{\sqrt{\mathbf{A^2 + B^2}}} \tag{5.4}$$

$$\mathbf{d} = \frac{\left|\mathbf{b_{superior} - b_{inferior}}\right|}{\sqrt{\mathbf{m^2 + 1}}} \tag{5.5}$$

Mediante a situação do corredor aplica-se assim a fórmula de cálculo das ordenadas na origem de r_1 e r_2 e das retas auxiliares em função da reta principal, de acordo com a fórmula 5.6.

$$\begin{cases}\mathbf{b_{superior} = \sqrt{1 + m^2} \times d + b_0} \\ \mathbf{b_{inferior} = b_0 - \sqrt{1 + m^2} \times d}\end{cases} \tag{5.6}$$

No fim, as retas auxiliares são igualadas à reta de meta r_3 para determinação das balizas de aproximação B_1 e B_2.

5.4.2.2. Cálculo das retas de estreitamento

Uma vez definidas as retas principais do corredor, a reta de meta e as balizas passa-se ao cálculo das retas de estreitamento r_4 e r_5.

As retas de estreitamento têm como função a definição de uma zona de convergência a fim de levar o veleiro a passar a reta de meta entre boias imaginárias, sendo estas definidas pelos pontos das balizas.

Uma estratégia possível para o cálculo destas retas será, à semelhança do que foi feito em eVentos4 [4], deduzir o declive da mesma pela fórmula do ângulo agudo entre duas retas, sendo isto feito a partir de cálculo tangencial entre a reta r_3 e as retas r_4 e r_5, método este ilustrado na figura 5.20.

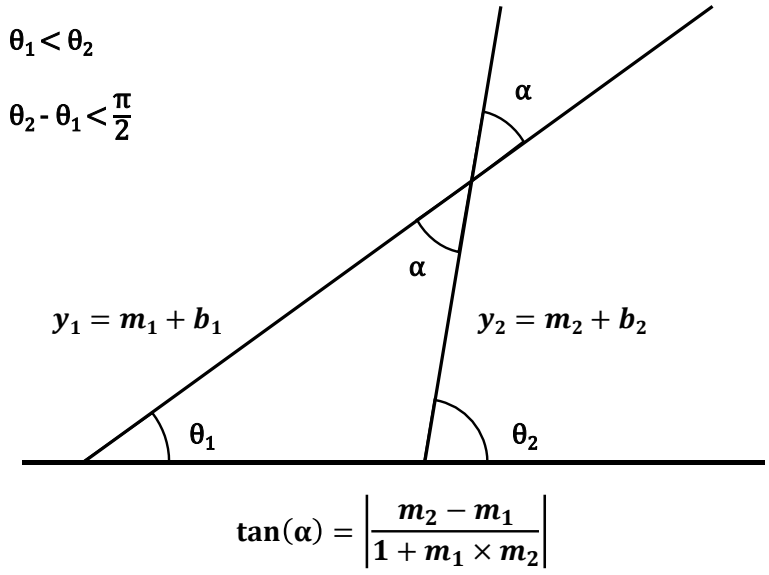


Figura 5.20: Método de cálculo tangencial do ângulo entre duas retas [4], [54].

Este método apresenta um problema: sendo que é realizado a partir dos valores dos declives das retas, este vai implicar que nenhuma delas seja vertical, já que neste caso o declive será indefinido. Como tal decidiu-se utilizar um método alternativo baseado em matrizes de rotação (fórmulas 5.7 e 5.8) [55].

Considerando os pontos das balizas de estreitamento B_1 e B_2, caso se queira calcular a reta r_4 roda-se o ponto B_2 em torno de B_1 no sentido anti-horário e caso se queira determinar r_5 roda-se B_1 em torno de B_2 no sentido horário.

No fim calcula-se a equação da respetiva reta a partir do ponto fixo de rotação e o novo ponto determinado.

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & -sen(\varphi) \\ sen(\varphi) & \cos(\varphi) \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{, sentido anti-horário} \qquad (5.7)$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & sen(\varphi) \\ -sen(\varphi) & \cos(\varphi) \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{, sentido horário} \qquad (5.8)$$

Isto é realizado pelo método auxiliar *rotateLine* que recebe as coordenadas do ponto fixo sobre o qual a rotação é desempenhada e o ponto a rodar. Este faz uma translação dos pontos em função da origem do referencial, subtraindo as coordenadas do ponto fixo de rotação ao ponto a rodar (rotação em torno da origem) e, mediante um parâmetro booleano, fará uma rotação ou no sentido anti-horário ("true") ou no sentido horário ("false"). No fim volta a somar as coordenadas inicialmente subtraídas ao ponto rodado e calcula ou o declive e a ordenada na origem da reta ou, caso a reta seja vertical, a longitude desta.

5.4.3. Cálculo da coordenada intermédia de bolina

De modo a que o veleiro consiga navegar à bolina dentro do corredor há a necessidade da definição de coordenadas de destino intermédias de modo a que este navegue alternadamente ao longo destas numa trajetória em ziguezague.

Para o efeito utilizou-se uma estratégia baseada naquela implementada em eVentos4 [4] que consiste no cálculo da reta paralela a r_0 que passa no ponto atual de localização do veleiro, seguida do cálculo da interseção desta com a reta r_3 e de uma rotação do ponto de interseção mediante um certo ângulo. No fim é verificado o ponto de interseção da reta resultante para com a reta de meta r_3, sendo esse o ponto de coordenada intermédia, conforme exemplificado na figura 5.21.

Este cálculo é feito pelo método *tackingCoord* que também faz uso do método *rotateLine* de matrizes de rotação, ao contrário do cálculo tangencial utilizado em eVentos 4.

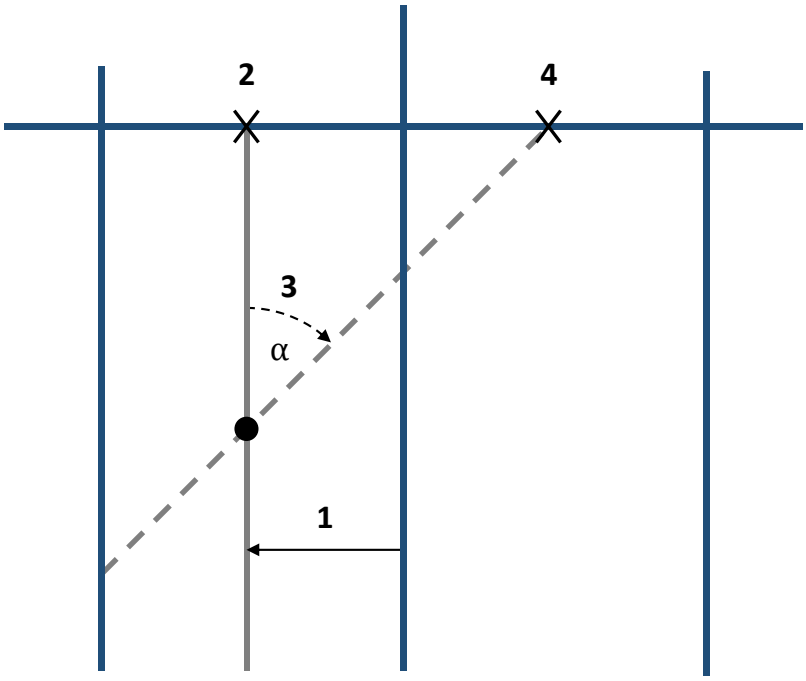


Figura 5.21: Diferentes passos do cálculo da coordenada intermédia de bolina baseados em eVentos 4 [4].

5.4.4. Cálculo da zona de localização no corredor

Para permitir a trajetória à bolina é necessário que a máquina de estados saiba em que zona dentro do corredor o veleiro se encontra a fim de alternar entre coordenadas intermédias. Para tal dividiu-se o corredor em 7 zonas distintas de navegação conforme indicado na figura 5.22.

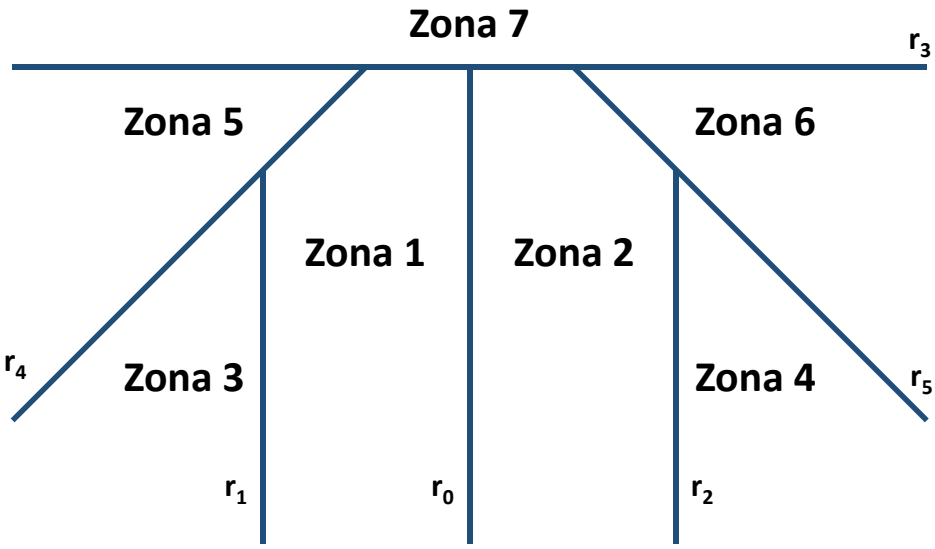


Figura 5.22: Zonas do corredor.

A divisão considerada pela dissertação eVentos 4 parte de dois cenários possíveis mediante a diferença longitudinal entre os pontos de destino e origem do corredor, cada qual com uma configuração diferente de zonas. Sendo que para o corredor desenvolvido nesta dissertação consideraram-se 8 situações possíveis de orientação, dada a eventualidade de uma das retas ser vertical, determinou-se que as zonas deste mantêm a sua posição relativa umas às outras para todas as situações, sem a necessidade da utilização de cenários.

Dependendo da situação em que se encontra o corredor, já determinada pelo método *calculateCorr*, começa-se por verificar por exclusão de partes qual a zona onde o veleiro se encontra atualmente. Em primeiro lugar verifica-se se este se encontra na zona 7 (além da linha de meta). De seguida verifica-se se está nas zonas 5 ou 6. Se não se encontrar em nenhuma destas parte-se de um extremo mediante a situação e verifica-se se está localizado numa das zonas de 1 a 4.

Isto é feito pela função *calculateZone* que recebe como parâmetros de entrada a longitude e latitude da posição corrente do veleiro, guardando a sua zona de localização no corredor na variável *zone*.

5.5. Nível 3

O nível 3 do controlador tem como função a definição das coordenadas de etapa constituintes do percurso de navegação assim como a determinação da condição de chegada a cada uma delas de modo a determinar o próximo destino a atingir. Este nível é definido pela classe *Level3* do código do controlador.

A definição das etapas do trajeto é feita a partir de um "array" de dimensão fixa ao qual, no início do funcionamento do controlador, se adiciona um certo número de pares longitude-latitude de coordenadas geográficas.

Para a determinação da condição de chegada analisaram-se duas estratégias de interesse disponíveis na linha de projetos eVentos: uma estratégia proposta pela dissertação eVentos 2 e outra implementada por eVentos 4.

A estratégia proposta pela dissertação eVentos 2 considera uma área circular com raio pré-definido centrada no ponto do próximo destino. Caso o veleiro se encontre dentro da dita área, o objetivo é considerado atingido, como é indicado na figura 5.23 [2]. A dissertação eVentos 4, por sua vez, define uma estratégia baseada no corredor de navegação à bolina, definindo que o objetivo é atingido caso o veleiro passe além da linha de meta r_3 do corredor [4].

Condição de chegada: $\sqrt{dx^2 + dy^2} < r$

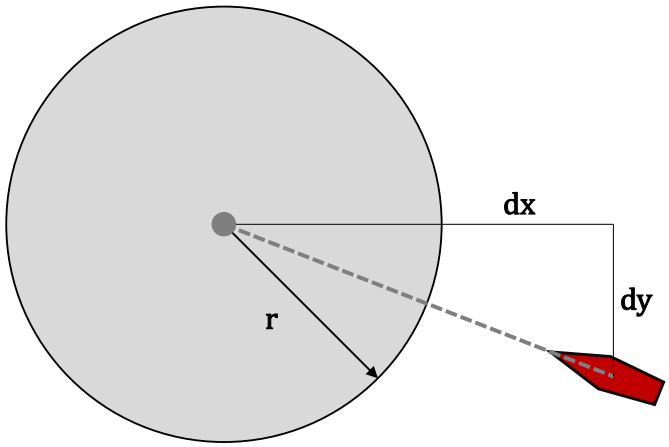


Figura 5.23: Condição de chegada proposta por eVentos 2 [2].

Sendo que a estratégia implementada por eVentos 4 pode levar a resultados imprecisos, caso a corrente leve o veleiro a cruzar a linha de meta além do intervalo entre as balizas de navegação, optou-se então pela estratégia proposta por eVentos 2, contudo no decorrer de alguns ensaios iniciais verificou-se um problema: apesar do GPS de modelo MT3339 ter uma precisão garantida pelo fabricante de 1.8 metros, na prática registaram-se várias falsas chegadas de etapa decorrentes de leituras imprecisas por parte do GPS.

A solução encontrada para este problema foi então a junção de um contador de deteção de chegada à etapa. A fim de uma etapa ser considerada atingida é necessário que o controlador detete o veleiro dentro da área circular um certo número de vezes (ciclos do controlador) e, quando o contador iguale ou ultrapasse o limite pré-estabelecido de deteções, a etapa é considerada atingida. De seguida, após a chegada ao objetivo atual, o nível 3 considera a próxima etapa no "array" como o destino a atingir.

5.6. Máquina de estados RdP

Além dos 3 níveis principais da estratégia do controlador é também necessário a presença de um componente que garanta a coordenação entre os diferentes estados em que este se possa encontrar. Para tal, este componente tem de ter conhecimento de três características de controlo:

- **Estado de bolina:** capacidade de detetar se o veleiro se encontra a navegar à bolina através de uma variável de entrada proveniente do nível 2, a fim de o controlador saber quando deve calcular o corredor de navegação;
- **Zona do corredor:** uma vez à bolina, o controlador deve saber em que zona do corredor o veleiro se encontra de forma a determinar se deve calcular a próxima coordenada intermédia, assim como o seu sentido;
- **Comutação entre modo autónomo e manual:** o controlador deve ser capaz de saber se se encontra ou em modo autónomo de navegação, ou em modo de controlo manual, a fim de selecionar os valores a enviar para os servos, entre o nível 1 e o sistema de controlo manual.

Uma solução implementada pela dissertação eVentos 4 a fim de atender aos primeiros dois pontos mencionados foi a implementação de uma RdP de três lugares, como descrito na figura 3.13 [4], contudo existem alguns pontos relevantes a considerar.

O corredor de bolina implementado na presente dissertação não necessita da definição de vários cenários pois, ao contrário do corredor desenvolvido pela dissertação eVentos 4, o corredor utilizado mantém a posição das retas e das zonas relativamente umas às outras, independentemente da posição do ponto de destino em função do ponto de origem. Além disso, a máquina de estados de eVentos 4 não define um estado relativo ao controlo manual decorrente da comutação efetuada pelo respetivo sistema periférico.

Atendendo a estas características procedeu-se à implementação de uma máquina de estados em notação de rede de Petri composta por 4 lugares e 10 transições feita a partir de uma extensão da rede definida por eVentos 4, obtendo-se a rede da figura 5.24.

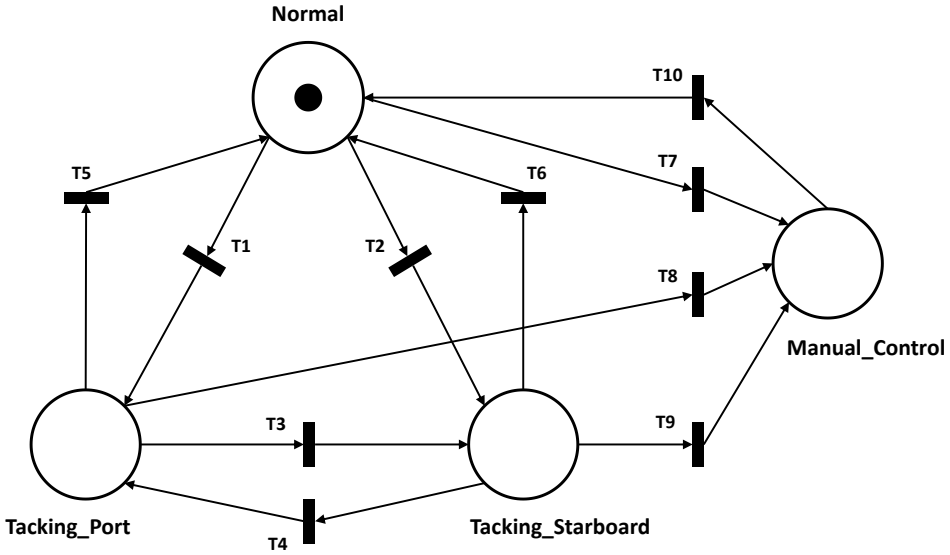


Figura 5.24: RdP implementada, baseada na rede de eVentos 4 [4].

Os lugares *Normal*, *Tacking_Port* e *Tacking_Starboard* correspondem ao modo autónomo de navegação, com o primeiro referente ao estado de navegação em condições de vento favorável e os dois últimos referentes à navegação à bolina. O estado *Manual_Control* corresponde ao modo de controlo manual.

Esta rede conta com três sinais de entrada correspondentes ao estado de bolina (*InS_Wind*), a zona do corredor (*InS_Zone*) e o estado de comutação entre os modos de controlo autónomo e manual (*InS_ManualCtrl*), de acordo com a tabela 5.8.

Tabela 5.8: Descrição dos sinais de entrada da rede de Petri.

Sinal	Tipo	Descrição
InS_Wind	Booleano	Estado de bolina (“true”)
InS_ManualCtrl	Booleano	Comutação entre controlo manual (“true”) e autónomo (“false”)
InS_Zone	Integer ϵ [1, 7]	Zona do corredor da localização corrente do veleiro

Como sinal de saída, a rede tem uma variável denominada *OuS_Place* que guarda um número de 1 a 4 associado a cada lugar que a constitui, como indicado na tabela 5.9.

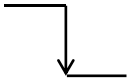
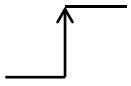
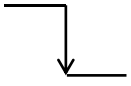
Tabela 5.9: Valores do sinal de saída da RdP em função do lugar atual.

Lugar	**Valor do sinal de saída "OuS_Place"**
Normal	1
Tacking_Port	2
Tacking_Starboard	3
Manual_Control	4

Os sinais do estado de bolina e da zona do corredor são dados pela classe *Level2* do controlador. O booleano de comutação entre os modos autónomo e manual é fornecido pela classe *Joystick_RCV*. O limite máximo de 7 da variável *InS_Zone* serve para abranger a zona além da linha de meta que, apesar de não ser utilizada, poderá vir a ser útil a iterações futuras do projeto.

De acordo com a tabela 5.10, são definidos três eventos de natureza "rising/falling-edge" em função dos sinais de entrada da rede. Estes serão responsáveis pelo disparo das transições da rede.

Tabela 5.10: Eventos da RdP.

Evento	**Sinal associado**	**Ocorrência**
IE_favorable	InS_Wind	 true → false
IE_manual	InS_ManualCtrl	 false → true
IE_autonomous	InS_ManualCtrl	 true → false

A partir dos eventos descritos e dos sinais *InS_Zone* e *InS_Wind* definiram-se as condições de disparo de cada uma das transições da rede, de acordo com a tabela 5.11.

Tabela 5.11: Condições de disparo das transições da RdP.

Transição	Condições de disparo	Prioridade
T1	InS_Wind = 1 AND (InS_Zone = 2 OR InS_Zone = 4 OR InS_Zone = 6)	2
T2	InS_Wind = 1 AND (InS_Zone = 1 OR InS_Zone = 3 OR InS_Zone = 5)	2
T3	InS_Wind = 1 AND (InS_Zone = 3 OR InS_Zone = 5)	2
T4	InS_Wind = 1 AND (InS_Zone = 4 OR InS_Zone = 6)	2
T5	Evento IE_favorable	2
T6	Evento IE_favorable	2
T7	Evento IE_manual	1
T8	Evento IE_manual	1
T9	Evento IE_manual	1
T10	Evento IE_autonomous	1

As transições associadas com a comutação entre os modos de controlo autónomo e manual (lugar *Manual_Control*) têm uma prioridade mais alta que as restantes (incluindo *T10* apesar de ausência de concorrência) devido ao facto de que esta poderá vir a ser ativada em situações de emergência.

As transições *T1* e *T2* ocorrem quando o veleiro se encontra em situação de vento desfavorável. Consoante a zona atual do veleiro dentro do corredor, o controlador irá calcular uma coordenada intermédia ou a bombordo (zonas 2, 4 ou 6 do corredor) ou a estibordo (zonas 1, 3 e 5). Durante a condição de bolina pretende-se que o veleiro mantenha a sua trajetória em ziguezague dentro das zonas 1 e 2 assim como dentro das retas de estreitamento.

Caso o controlador esteja no estado *Tacking_Port* e entre nas zonas 3 ou 5 a transição *T3* dispara a fim de o controlador calcular uma nova coordenada intermédia a estibordo. Caso esteja no estado *Tacking_Starboard* e entre nas zonas 4 ou 6 a transição *T4* dispara a fim de calcular uma nova coordenada a bombordo. Isto garante que o veleiro faça a sua trajetória em ziguezague. As transições *T5* e *T6* ocorrem quando o veleiro já não se encontre em situação de bolina, voltando ao estado *Normal*.

5.7. Sistema de controlo manual

No decorrer de uma situação de navegação pode ser útil permitir ao utilizador comutar a estratégia autónoma de navegação para um modo de controlo manual. Para o efeito criou-se um sistema periférico composto por um "joystick shield" ITEAD assente num Arduino MEGA 2560 extra.

Este sistema comunica com o Arduino do controlador em modo unicast por intermédio de dois módulos rádio APC220 (um para cada microcontrolador) que por sua vez comunicam com os respetivos microcontroladores por protocolo série.

5.7.1. Trama de informação

A fim de se garantir a comunicação entre o Arduino emissor detentor do "joystick" e o Arduino do controlador considerou-se que se precisa, no mínimo, de 26 bits: 6 bits para o estado dos botões e 10 bits para cada eixo do joystick adquiridos por leitura analógica (1024 valores possíveis para cada eixo).

Uma vez que a unidade elementar de envio pelos métodos *Serial.write()* e *Serial.read()* é o byte, compôs-se uma trama de informação de 4 bytes. Esta trama é construída a partir de "flags" binárias e operações "OR" e "AND" de acordo com os valores lidos do joystick.

Como a comunicação é assíncrona, há que garantir que o Arduino recetor consiga distinguir uma trama completa do resto da informação do "buffer". Para tal optou-se por deixar o bit mais significativo de cada um dos quatro bytes de informação a zero e adicionou-se um byte extra de preâmbulo com o valor 255 (impossível de se obter nos bytes de informação), a fim de o recetor saber qual é o início da trama, ficando-se assim com um total de 5 bytes (1 de preâmbulo mais 4 de informação) como apresentado na figura 5.25.

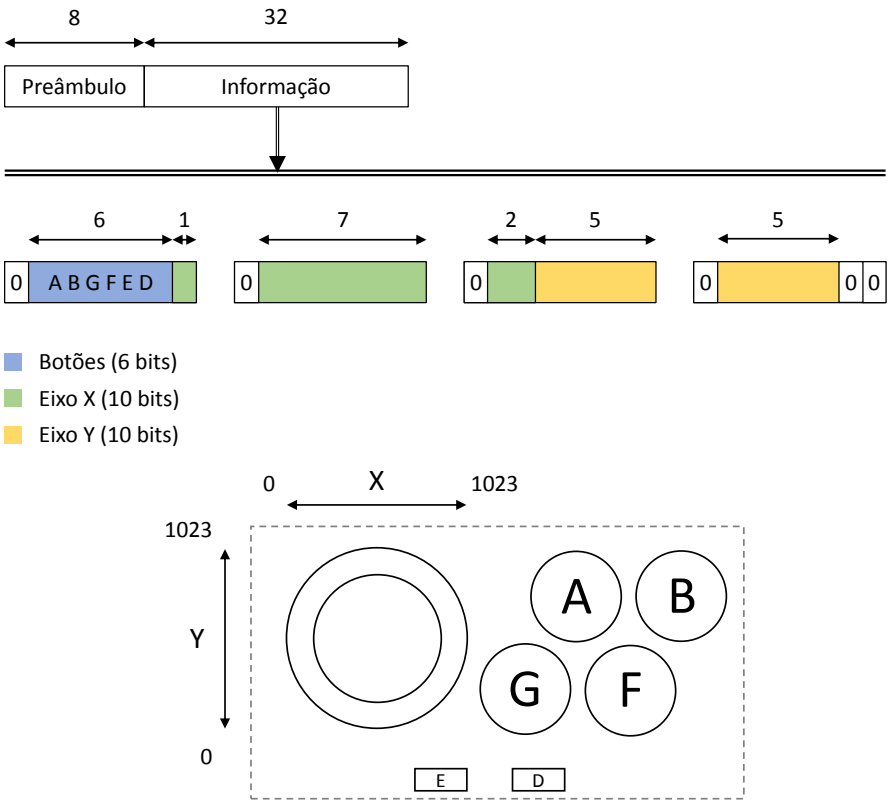


Figura 5.25: Composição da trama do sistema de controlo manual *em cima*; Eixos e botões do joystick *em baixo*.

De notar que os dois bits menos significativos do último byte de informação ficam sempre a zero já que não serão necessários, não se precisando assim de forçar o primeiro bit a zero, contudo decidiu-se mantê-lo na mesma forma dos restantes. Além disso, apesar de a comutação só necessitar de um botão (botão "A" neste caso), optou-se por incluir informação na trama referente a todos os botões, garantindo assim uma "template" que pode ser útil a projetos futuros.

5.7.2. Estratégia de controlo

A fim de levar o manípulo do joystick a mudar a posição dos servos da vela e do leme considerou-se uma estratégia em que, uma vez determinados os valores centrais de cada eixo, definiu-se uma pequena margem em torno dos mesmos dentro da qual o Arduino recetor não tomará ação nos servos, isto devido a pequenas oscilações de valor provenientes de movimentos indesejados do manípulo ou de desgaste, o que levaria os servos a moverem-se sem a devida intenção do utilizador. Fora dessas margens, dependendo da proximidade do manípulo para com o extremo de cada eixo, aumentará ou diminuirá o valor de incremento ou de decremento a aplicar ao valor PWM dos servos, como indicado na figura 5.26.

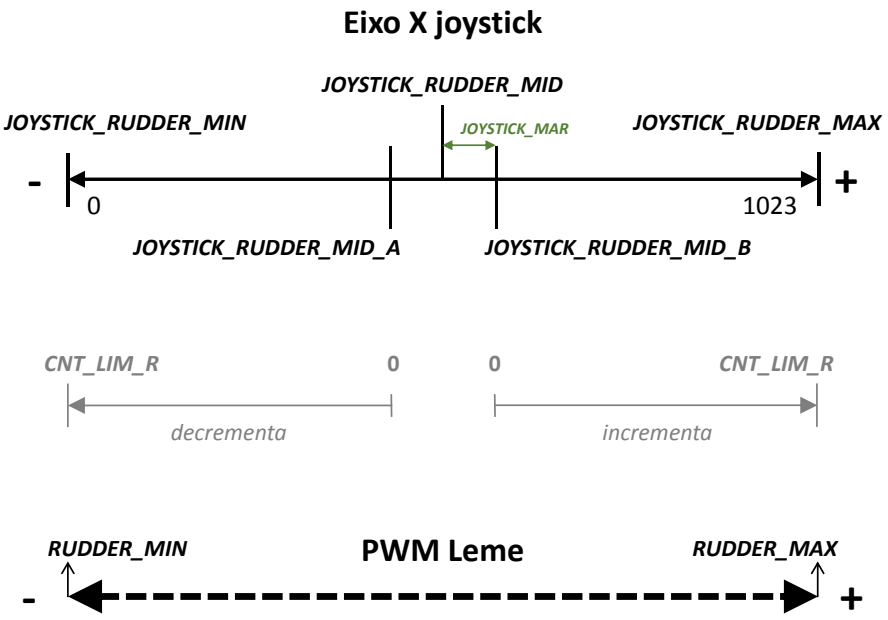


Figura 5.26: Estratégia de controlo do eixo horizontal do joystick *em cima* e efeito resultante na escala do leme *em baixo*.

No decorrer da calibração determina-se o valor central do eixo deixando este parado, sendo esse valor igualado à constante *JOYSTICK_RUDDER_MID*, assim como os extremos do eixo horizontal (*JOYSTICK_RUDDER_MIN* e *JOYSTICK_RUDDER_MAX*).

Em torno do valor central definem-se os limites de uma margem, subtraindo e somando o valor *JOYSTICK_MAR*. Dentro desta margem o controlo manual não terá efeito.

À medida que o utilizador move o manípulo para a esquerda, o valor PWM do servo irá decrementar ciclo a ciclo um número de unidades que vai de 0, na posição *JOYSTICK_RUDDER_MID_A*, ao valor máximo de *CNT_LIM_R* unidades, quando na posição mínima *JOYSTICK_RUDDER_MIN*. Quando o manípulo é movido para a direita irá ocorrer um incremento que pode ser de 0 unidades, em *JOYSTICK_RUDDER_MID_B*, a *CNT_LIM_R* unidades na posição *JOYSTICK_RUDDER_MAX*. O controlo do servo da vela é feito de maneira semelhante para caçar/folgar a vela, correspondendo aos movimentos para baixo e para cima do eixo vertical respetivamente, de acordo com a figura 5.27.

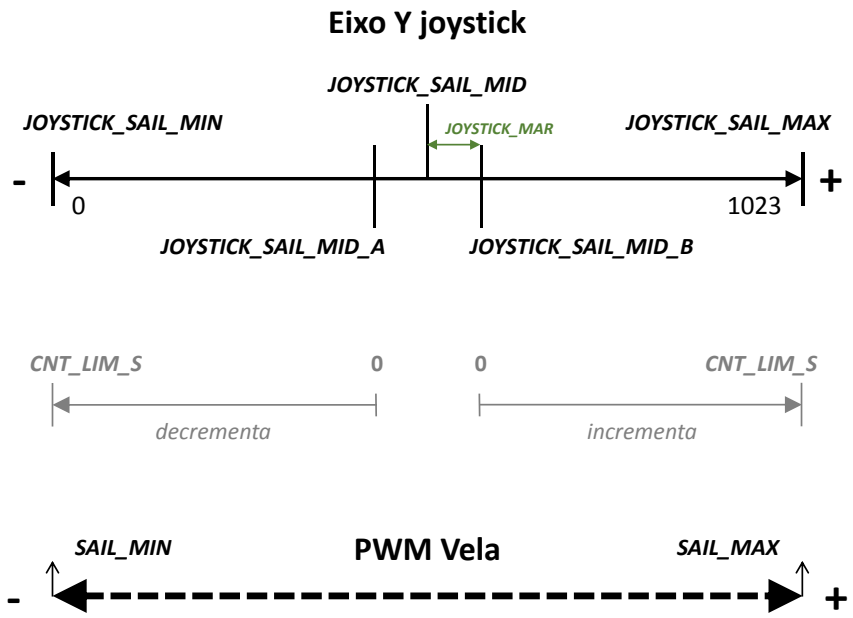


Figura 5.27: Estratégia de controlo do eixo vertical do joystick *em cima* e efeito resultante na escala da vela *em baixo*.

Os limites máximos de incremento e decremento de cada eixo (constantes *CNT_LIM_R* e *CNT_LIM_S*) são determinados durante a calibração, consoante a velocidade de resposta de cada servo.

5.7.3. Estratégia de comutação

Antes de o utilizador poder controlar o veleiro diretamente pela utilização do joystick há que definir uma estratégia que permita a comutação entre os modos de navegação autónoma e de controlo manual. Dada a natureza assíncrona da comunicação, o pior caso possível de receção ocorre quando o controlador recebe a trama de informação do sistema de controlo manual a partir do segundo byte da mesma, pelo que este deverá esperar por receber pelo menos 2×N-1 bytes de informação, com N igual ao número total de bytes de uma trama completa, ou seja, 9 bytes de informação. Uma vez recebidos estes 9 bytes, o controlador irá de seguida detetar o valor de início da trama (255) e de seguida vai ver se o bit correspondente ao botão de comutação tem o valor 1 (botão "A").

A fim de impedir passagens bruscas entre os modos de controlo, ou mudanças não desejadas por erro de transmissão, definiu-se uma estratégia baseada num contador em que a comutação só ocorre caso tenham sido detetadas um certo número de tramas consecutivas em que o botão de comutação tenha sido pressionado, contudo devido ao facto de que os tempos de ciclo variam entre as condições de vento favorável e desfavorável, assim como devido a diferenças de tempo na aquisição de valores por parte do GPS, é difícil achar um valor para o contador que se adequa a todas as situações possíveis.

A solução passou por juntar ao contador um "timer" que define um tempo de intervalo mínimo entre comutações. Assim, uma vez feita uma comutação, o número de tramas com o bit do botão "A" com valor "true" só voltam a ser contabilizadas quando um certo intervalo de tempo ter passado.

Para o efeito escolheu-se um número limite baixo como valor do contador (*COMUTE_LIM = 3*) com um tempo mínimo entre comutações de 4 segundos (constante *TIME_MODE*), ambos definidos no ficheiro "Consts.h".

5.8. Aplicação de monitorização

Para ser possível comprovar o funcionamento do controlador e de informar o utilizador do seu estado atual é necessário ter um sistema capaz de registar os valores provenientes do controlador e de fornecer feedback em tempo real do estado de navegação. Para tal elaborou-se uma aplicação de monitorização em linguagem Java com recurso à "framework" Swing para o desenvolvimento de uma interface gráfica, por utilização da biblioteca externa jSerialComm [56] para comunicação com protocolo série.

A aplicação é usada num computador que comunica com o Arduino do controlador por intermédio de um par de módulos APC220 à semelhança do controlo manual, utilizando-se um conversor TTL/USB para ligar um módulo a uma porta USB do PC. A comunicação é feita em modo unicast com a aplicação no papel de recetor e o controlador no papel de emissor.

O GUI da aplicação conta com 4 elementos principais (figura 5.28):

- **Selecionador de "serial port" de comunicação:** situado no topo do GUI que consiste numa "combo box" e botão "Start" com a lista de portas série disponíveis para comunicação;
- **Painel "Navigation Status":** painel com informação da orientação do próximo destino, ângulo de correção, valores dos servos do leme e da vela e valor do "roll";
- **Painel "Sensors":** informação da bússola, cata-vento (em escalas simétrica e assimétrica) e coordenadas atuais do GPS;
- **Informação da RdP e lista de etapas:** situado no lado direito, consistindo numa imagem com informação do estado atual da RdP e uma tabela indicativa da lista de coordenadas do trajeto com realce da coordenada atual a atingir.

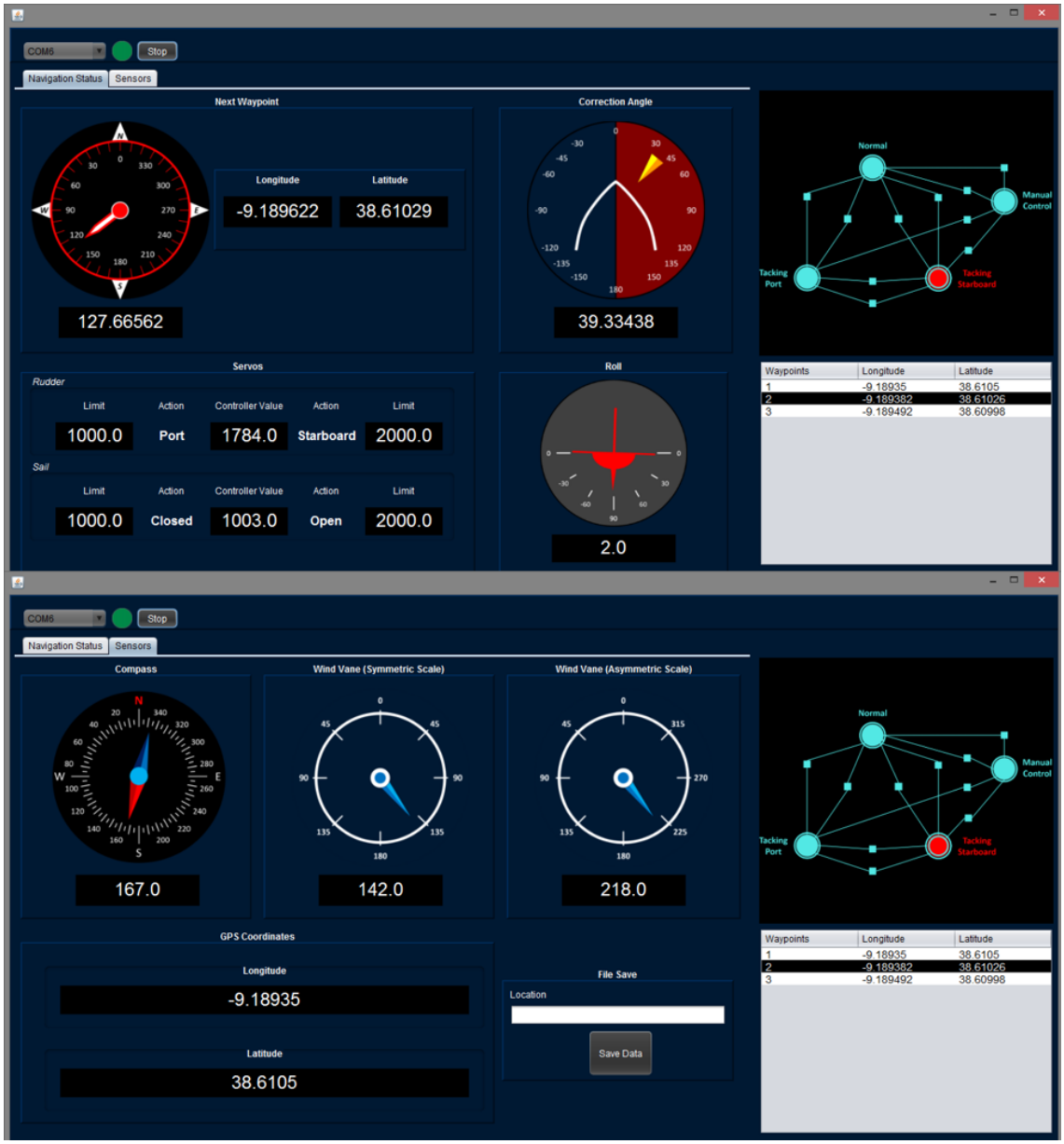


Figura 5.28: Diferentes painéis do GUI da aplicação de monitorização.

5.8.1. Trama de informação

Uma característica a ter em conta quanto à comunicação entre o Arduino do controlador e o PC é o seu caráter assíncrono. A utilização de um protocolo que envolva "handshaking" não é a solução mais indicada já que resultaria em ciclos mais lentos por parte do controlador devido à confirmação necessária de ambas as partes no estabelecimento da ligação.

Devido a este facto, optou-se por um protocolo de datagramas simples a enviar em modo unicast com uma trama de informação composta em binário.

A fim de o recetor conseguir distinguir uma trama completa de qualquer combinação de tramas sucessivas presentes no "buffer" consideraram-se dois valores como sendo os indicadores de início e fim da trama, tendo o byte indicador de início o valor 255 ("unsigned byte") e o byte indicador de fim o valor 254. Dado que estes valores podem-se confundir com os próprios bytes de informação, caso estes contenham valores semelhantes, considerou-se que cada ocorrência destes valores ao longo da trama é sempre sucedida de um segundo byte que pode tanto tomar o valor 0, caso suceda a um limitador, ou o valor 1 caso suceda a um byte de informação.

Outro problema a ter em conta é a diferença de dimensão entre os tipos primitivos das variáveis utilizadas no microcontrolador e aquelas utilizadas no computador. No microcontrolador Arduino MEGA 2560, um "double" é exatamente igual a um "float" em Java do standard IEEE-754 [57] (32 bits), devendo-se isto às limitações da arquitetura da plataforma e ao facto de o IDE ser utilizado por diferentes tipos de microcontroladores Arduino, alguns dos quais suportando o standard comum de 64 bits para o tipo primitivo "double". Não obstante, manteve-se a maioria das variáveis do controlador como "double" a fim de permitir a obtenção de maior precisão caso ocorra a migração do código do controlador para uma plataforma mais avançada em projetos futuros.

Tem-se assim uma trama de informação constituída pelos valores dos diferentes sensores, atuadores, informação do corredor, estado da RdP e informação relevante às coordenadas do trajeto, como indicado na figura 5.29.

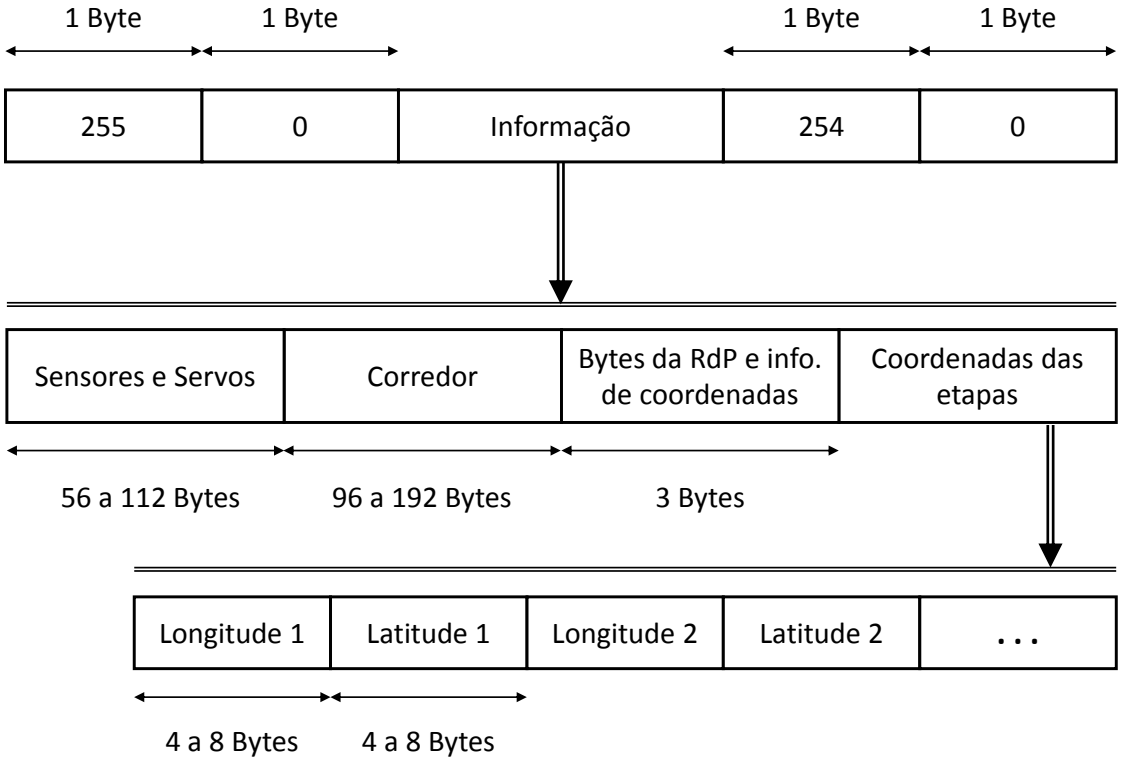


Figura 5.29: Dimensão da trama de informação da aplicação de monitorização.

Dada a possibilidade de uma duplicação total das variáveis do tipo "float" enviadas, isto devido à eventualidade de cada byte coincidir ou com o valor 255 ou 254, estas podem ter uma dimensão de 4 a 8 bytes devido à inclusão do byte sucessor indicador de informação (valor 1). Este só é adicionado se tal caso se verifique.

Além dos valores referentes aos respetivos sensores e atuadores, a trama também contém informação detalhada do corredor, tais como os declives das retas (m_n), valores da ordenada na origem das mesmas (b_n) assim como, caso se verifique, longitudes das retas verticais (vX_n) e também as coordenadas das balizas de estreitamento. Para o registo do estado atual da RdP, o número da próxima etapa do percurso e o número total de etapas utilizam-se 3 bytes individuais, estando a ordem de todos estes valores descrita na tabela 5.12.

Devido à ausência de coleções dinâmicas nas bibliotecas nativas do Arduino, preparou-se a aplicação para um máximo de 100 pares de coordenadas de etapa já que tal número é mais que suficiente para testes de navegação.

Tabela 5.12: Constituintes da trama de informação da aplicação de monitorização.

Ângulo de destino	Ângulo de correção	Lim. mínimo do servo do leme	Lim. máximo do servo do leme	Servo do leme
Lim. mínimo do servo da vela	Lim. máximo do servo da vela	Servo da vela	"Roll" da bússola	"Bearing" da bússola
Cata-vento (simétrico)	Cata-vento (assimétrico)	Longitude GPS	Latitude GPS	Longitude do próximo destino
Latitude do próximo destino	Longitude da origem do corredor	Latitude da origem do corredor	m0	b0
vX0	b1	vX1	b2	vX2
m3	b3	vX3	m4	b4
vX4	m5	b5	vX5	Longitude de B1
Latitude de B1	Longitude de B2	Latitude de B2	Lugar da RdP	N.º da próxima etapa
N.º de etapas	Longitude 1	Latitude 1	Longitude 2	…

5.8.2. Funcionamento

A base da aplicação consiste em dois "timers": um evocado de 5 em 5 segundos que é responsável por fazer um update das portas série disponíveis e apresentá-los na respetiva "combo box" e um segundo evocado de 250 em 250 milissegundos responsável pelo processamento de dados, update do GUI e do guardar dos valores do controlador numa string, sendo este o timer principal.

Assim que o utilizador seleciona a porta série de comunicação onde se liga o módulo RF recetor e pressiona o botão "Start", a aplicação dará início ao funcionamento do timer principal. Este vai por sua vez evocar uma "worker thread" que iniciará o processamento da trama recebida por evocação do método *processData* da classe *COM*.

Caso tenham passados pelo menos *TIMER_SAVE_VALUES* milissegundos desde a última evocação da "worker thread" (constante de amostragem), esta também irá evocar a função *setFileData* que irá guardar numa string os dados atuais processados referentes ao controlador. No fim destas ações a "worker thread" irá fazer update do GUI, fazendo "schedule" para a EDT.

Se o utilizador quiser guardar os dados em formato ".txt" basta ir ao painel "Sensors" e clicar no botão "Save Data" a fim de escolher a diretoria onde guardar o ficheiro com os resultados do controlador até então obtidos.

5.9. Resumo de melhorias efetuadas

O controlador desenvolvido resultou em melhorias feitas a todos os níveis de controlo definidos pelas dissertações eVentos 3 e eVentos 4 assim como na criação de dois novos dispositivos periféricos e no aproveitamento de alguns conceitos definidos por eVentos 2.

O nível 1 do corredor foi refeito a partir de um misto das funções pertença definidas por eVentos 3, assim como da junção da variável de controlo do "Roll" a fim de garantir mais estabilidade ao veleiro em contextos de navegação.

O corredor do nível 2 foi calculado de maneira diferente a fim de se obter uma estratégia mais robusta capaz de lidar com possíveis casos de retas verticais e assimetrias das distâncias entre retas, balizas e ângulos das retas de estreitamento, a fim de garantir mais versatilidade em possíveis contextos de navegação e maior precisão no cálculo das distâncias entre as retas.

O nível 3 foi implementado a partir de uma sugestão dada pela dissertação eVentos 2. Isto foi feito em alternativa à estratégia de cruzamento da linha de meta do corredor (r_3), pois esta última requer o cálculo do corredor a cada ciclo (o que resulta em ciclos mais lentos) e pode ser imprecisa caso o efeito da corrente desvie o veleiro para além das balizas de aproximação.

A rede de Petri resulta de uma extensão da rede definida por eVentos 4 a fim de albergar o estado de controlo manual, assim como na redução de uma variável: o cenário do corredor.

O sistema de controlo manual e a aplicação de monitorização foram criadas a fim de se obterem meios capazes de proporcionar tanto o controlo direto como o registo de resultados para confirmação da performance do controlador.

Sumarizando os melhoramentos feitos ao controlador obteve-se os pontos referidos na tabela 5.13.

Tabela 5.13: Resumo de melhorias realizadas em eVentos 5.

Nível 1	Adição da variável "Roll" ao controlador difuso, sugerida por eVentos 3.
Nível 2	Método alternativo de cálculo do corredor implementado por eVentos 4 com melhorias a vários níveis: • Verificação de situações de verticalidade das retas; • Cálculo da distância entre retas pela equação da distância entre retas paralelas; • Assimetria das balizas de aproximação, ângulos de estreitamento e das retas limite do corredor; • Cálculo das retas de estreitamento e coordenadas intermédias com recurso a matrizes de rotação.
Nível 3	Implementação de estratégia de chegada baseada em área circular, sugerida por eVentos 2.
Máquina de estados	Extensão da RdP de eVentos 4 para quatro estados, incluindo um estado de comutação para controlo manual.
Controlo manual	Desenvolvimento de protocolo de comunicação assíncrona para controlo manual via canal série, utilizando um "joystick shield" e dois módulos de comunicação rádio.
Aplicação de monitorização	Desenvolvimento de uma nova aplicação Java pelo uso da "framework" swing para monitorização e armazenamento de resultados, assim como a elaboração de um protocolo de comunicação.

Outra vantagem obtida pelo controlador desenvolvido foi a organização modular do código do sistema, dando maior facilidade à sua reutilização completa ou parcial em outros projetos, assim como o facilitar da alteração de valores referentes à calibração do sistema.

Além dos pontos anteriormente mencionados, também se elaborou um manual laboratorial contendo documentação em falta. Este documento conta com os seguintes conteúdos [58]:

- **Esquemas de ligação dos veleiros:** identificando as ligações dos fios presentes nas caixas de ligações de cada veleiro;
- **Esquemas de ligação e montagem dos sensores:** informação dos pinos a ligar, tanto dos sensores como do Arduino, assim como exemplos de código relativos a cada componente;
- **Passos de calibração:** guia passo a passo a fim de se obter os valores necessários ao bom funcionamento de cada componente do sistema;
- **Guia de utilização do software XFuzzy:** descrição dos princípios da lógica difusa assim como um guia de elaboração de um sistema difuso simples, mostrando os passos da criação de um sistema, ferramentas de análise e geração de código em linguagem C;
- **Listagem de funções de algumas bibliotecas úteis:** descrição de algumas funções das bibliotecas nativas do Arduino relativas a cada protocolo de comunicação utilizado, assim como uma listagem de alguns métodos relativos a bibliotecas externas de caráter auxiliar (ex. GPS).

Este manual tem como função auxiliar outros projetos, quer de dissertações externas, quer de estágios académicos que envolvam a utilização dos meios mencionados [59].

Capítulo 6

Validação e análise de resultados

Este capítulo faz uma análise dos resultados obtidos a partir dos diferentes testes realizados ao longo do desenvolvimento do controlador, descrevendo os procedimentos e ferramentas utilizadas na sua elaboração, assim como uma análise da performance de controlo em função dos objetivos requeridos.

O teste do funcionamento do controlador e seus constituintes foi feito ao longo de duas fases. Numa primeira fase, no decurso do desenvolvimento do código de controlo, foram feitos testes individuais a cada componente do sistema a fim de garantir que estes funcionam de acordo com o comportamento esperado antes da sua integração no sistema final. Para tal utilizaram-se várias ferramentas tais como o IDE do próprio Arduino (por verificação de output pela consola), a ferramenta de monitorização de resultados do XFuzzy para testes referentes ao nível 1, "scripts" de teste em linguagem C++ e a plataforma de desenho gráfico desmos para testes do corredor de navegação do nível 2, como no exemplo indicado na figura 6.1.

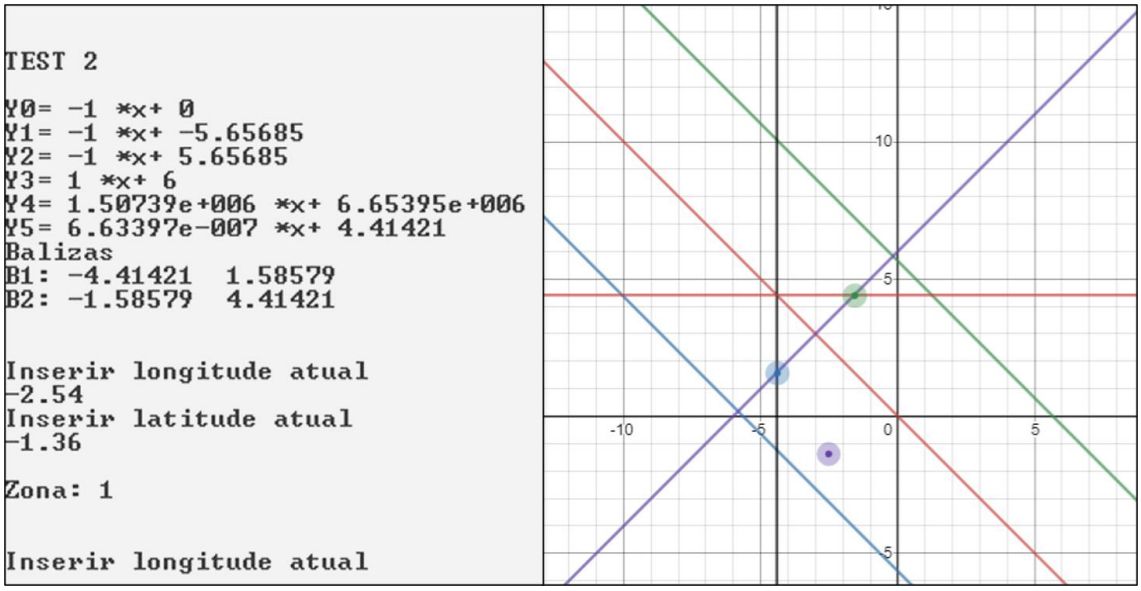


Figura 6.1: Exemplo de resultados do código de teste do corredor na consola *à esquerda* com gráfico na plataforma desmos *à direita* [32].

Após se completar a construção do controlador realizou-se uma segunda fase de testes no exterior para verificação do funcionamento do sistema:

- **Teste dos níveis 1 e 3:** este teste teve como função verificar a capacidade do veleiro em se orientar face a cada etapa de um trajeto pré-definido, assim como a deteção da chegada a cada uma delas;
- **Teste do nível 2:** teste que teve como função verificar a capacidade do controlador em definir um corredor de navegação, assim como o cálculo de coordenadas intermédias de bolina e a mudança de estado da RdP mediante a zona de localização do sistema dentro do corredor;
- **Teste completo:** teste misto para verificação do funcionamento do sistema na sua plenitude.

Devido a restrições derivadas da pandemia COVID-19, não foi possível a realização de testes num contexto de navegação.

Para a elaboração destes testes escolheram-se dois terrenos públicos longe de edifícios nos quais se marcaram diferentes pares de coordenadas definidoras das etapas de trajeto. A fim de emular uma situação de navegação, montou-se o sistema numa caixa de cartão que por sua vez é transportada em cima de um computador portátil. Este último corre a aplicação de monitorização que tem como objetivo guardar os valores das variáveis referentes ao desempenho do controlador assim como de fornecer feedback em tempo real, permitindo ao caminhante corrigir a sua orientação face ao próximo destino, quer seja esta uma etapa principal do percurso, quer seja uma etapa intermédia de bolina, por leitura do valor do ângulo de correção.

6.1. Teste dos níveis 1 e 3

O primeiro teste teve como objetivo testar a viabilidade do controlador em se orientar face às etapas de um percurso. Para tal definiu-se um percurso com 4 etapas (tabela 6.1), cabendo ao utilizador caminhar ao longo destas mediante o feedback prestado pela aplicação de monitorização.

Tabela 6.1: Etapas do trajeto do teste dos níveis 1 e 3.

Etapa	Longitude	Latitude
1	-9.189443	38.610533
2	-9.189368	38.610235
3	-9.190058	38.610386
4	-9.189443	38.610533

Antes da realização deste teste procedeu-se à calibração do cata-vento, assim como na definição dos valores relativos aos extremos dos servos e os diferentes parâmetros de comunicação da aplicação de monitorização, de acordo com a tabela 6.2.

Tabela 6.2: Valores de calibração do teste dos níveis 1 e 3.

Valores de calibração		
Cata-vento	PWM mínimo	705
	PWM máximo	3140
	Ângulo de compensação (valor à proa em graus)	132
Nível 3	Contador de chegada à etapa	3
	Raio circular de etapa (m)	8
Aplicação	Tempo mínimo entre envio de tramas do emissor (ms)	250
	Tempo mínimo entre *worker threads* da aplicação (ms)	250
	Constante de amostragem de registo (ms)	5000
	Baud Rate (bit/s)	19200
	Word (unidade elementar de transmissão em bits)	8
	Bits de paridade (controlo de erro)	1 (par)
Servo do Leme	PWM mínimo (bombordo)	1000
	PWM máximo (estibordo)	2000
Servo da vela	PWM mínimo (vela caçada)	1000
	PWM máximo (vela folgada)	2000

O teste consiste em percorrer as diferentes etapas pela respetiva ordem numérica, voltando no fim à etapa inicial do percurso (etapas 1 e 4 coincidentes) na ordem 1→2→3→1, estando as etapas indicadas na figura 6.2.

Figura 6.2: Etapas do trajeto do teste dos níveis 1 e 3 do controlador (Google Earth, imagem de 27/6/2007) [60].

Feito o teste, fez-se uma análise dos valores registados pela aplicação utilizando o software Google Earth com edição de ficheiros ".kml" para verificação da trajetória, obtendo-se os resultados da figura 6.3.

Figura 6.3: Trajeto do teste dos níveis 1 e 3 do controlador com áreas aproximadas de etapa *a azul* e pontos de partida/chegada às mesmas *a amarelo* (Google Earth, imagem de 27/6/2007) [60].

A tabela 6.3 contem alguns valores relevantes registados pela aplicação, nomeadamente o ângulo de correção, os valores dos servos do leme e da vela, o "roll" (com valores negativos para inclinação a bombordo e positivos para estibordo), o valor da bússola, o valor do cata-vento, coordenadas do GPS e número da próxima etapa no intervalo [0, n]. A partida da primeira etapa assim como a chegada às restantes estão indicadas nas linhas coloridas.

Tabela 6.3: Alguns valores registados pela aplicação de monitorização do teste dos níveis 1 e 3.

Correction angle	Rudder servo	Sail servo	Roll	Bearing	WV sym.	GPS lng	GPS lat	Next WP
-3.676178	1453	1489	0	190	91	-9.189438	38.61052	1
-7.5783386	1394	1683	0	186	66	-9.189423	38.61046	1
-1.3027344	1482	1697	1	191	64	-9.189407	38.610413	1
-5.7254944	1426	1697	-1	189	64	-9.189407	38.610382	1
-17.202408	1250	1997	3	178	12	-9.189392	38.61032	1
110.4357	1997	1425	1	191	98	-9.189392	38.610275	2
28.160622	1750	1350	0	104	109	-9.189438	38.61023	2
-7.760994	1390	1003	0	67	151	-9.189484	38.61023	2
-4.086975	1448	1003	0	74	173	-9.189606	38.61029	2
13.959419	1750	1003	0	97	161	-9.1896515	38.610336	2
-11.059967	1263	1003	0	73	155	-9.189728	38.61035	2
-2.9432602	1462	1003	0	83	155	-9.189789	38.610367	2
-8.948822	1361	1003	1	80	154	-9.18985	38.610382	2
-1.4556046	1480	1003	0	93	158	-9.189911	38.610397	2
-9.281494	1350	1003	0	98	160	-9.189972	38.610413	2
-17.769325	1250	1003	0	98	157	-9.190002	38.610413	2
166.60333	1997	1003	-8	88	178	-9.190048	38.610413	3
-9.683258	1335	1489	0	272	91	-9.190033	38.610413	3
-10.425873	1302	1433	0	271	97	-9.189896	38.610443	3
-5.01474	1436	1589	0	276	80	-9.189835	38.61046	3
-0.94329834	1487	1450	0	280	95	-9.189758	38.610474	3
-4.626404	1441	1521	0	275	88	-9.189713	38.61049	3
-4.313141	1445	1540	0	274	86	-9.1896515	38.610504	3
8.433716	1624	1459	1	285	94	-9.189575	38.61052	3
8	1614	1468	0	278	93	-9.189529	38.610535	3
6	1577	1450	0	276	95	-9.189514	38.610535	3
4.1729126	1552	1396	-1	264	96	-9.189453	38.610565	N/A

Analisando os valores na ferramenta de monitorização do XFuzzy confirmou-se que todos os pontos com exceção do último vão de acordo com o comportamento esperado. Como o último ponto corresponde ao fim do trajeto, o controlador deixa de funcionar (ponto irrelevante).

Após a análise da trajetória notou-se duas coisas. Em primeiro lugar a orientação face ao ponto exato das etapas tem um pequeno desvio, o que já seria de esperar dadas as imprecisões inerentes aos sensores, motivo pelo qual se optou pela estratégia de área circular para o nível 3.

Em segundo lugar verificou-se um ajuste lento do ângulo de correção após a chegada à etapa 2, notando-se também algumas correções abruptas no trajeto 2→3. Isto pode-se dever a vários fatores:

- **Interferência magnética entre sensores:** os campos magnéticos emitidos pela bússola e o GPS podem causar interferência mútua devido à sua proximidade dentro da caixa, assim como dos módulos RF utilizados para comunicação entre o controlador e a aplicação que, apesar de não necessários a este teste, foram na mesma utilizados para testar a sua capacidade de envio wireless de dados;

- **Existência de obstáculos:** a existência de uma árvore próxima da etapa 2 poderá ter causado dificuldades à receção de dados por parte do GPS pois, apesar desta ter uma altura reduzida, pode na mesma influenciar a qualidade de comunicação devido ao ângulo de posicionamento dos satélites face ao sistema;

- **Erros pontuais na comunicação:** por vezes a comunicação poderá sofrer uma falha pontual devido a erros de transmissão, sendo estes resultados filtrados pela aplicação;

- **Imprecisões do GPS:** o GPS pode ser suscetível a imprecisões, mesmo em terreno desobstruído e condições climatéricas favoráveis, o que pode levar a algumas irregularidades na trajetória.

Apesar de todos estes fatores, o sistema é capaz de levar um caminhante a deslocar-se ao longo de um trajeto de etapas pré-definidas. Sendo que o teste é realizado por ação humana meramente a partir do feedback visual da aplicação e que a ação do controlador será ainda mais célere e precisa, considerou-se o teste como bem sucedido.

Este código, referente aos níveis 1 e 3 juntamente com o código dos sistemas periféricos, ocupa 10% da memória flash do Arduino e 58% da RAM (sistema preparado para um máximo de 100 pares de coordenadas de etapa), registando um tempo de ciclo aproximado entre os 541 e os 543 milissegundos, não contando com possíveis atrasos que possam ocorrer da aquisição do GPS.

6.2. Teste do nível 2

O segundo teste consistiu na análise da capacidade do veleiro de navegar à bolina, verificando o cálculo do corredor, etapas intermédias e estados da RdP.

Para tal, definiram-se três etapas de percurso, apresentadas na tabela 6.4.

Tabela 6.4: Etapas do trajeto do teste do nível 2.

Etapa	Longitude	Latitude
1	-9.189718	38.616935
2	-9.189513	38.616675
3	-9.189718	38.616935

Antes deste teste definiram-se as características do corredor de navegação à bolina tais como as distâncias entre retas, balizas e ângulos de estreitamento, de acordo com a tabela 6.5.

Os restantes parâmetros não mencionados têm valores idênticos àqueles apresentados na tabela 6.2 do teste anterior, tendo-se mudado apenas o raio de etapa do nível 3 de 8 para 6 metros.

Tabela 6.5: Valores de calibração do teste do nível 2.

Valores de calibração			
Nível 3		Contador de chegada à etapa	3
		Raio circular de etapa (m)	6
Nível 2	*Corredor*	Distância *D1* (m)	6
		Distância *D2* (m)	6
		Distância *D3* (m)	2
		Distância *D4* (m)	2
		Ângulo de estreitamento θ_1 (graus)	45
		Ângulo de estreitamento θ_2 (graus)	45
	Bolina	Ângulo de coordenada intermédia (graus)	45
	Zona Proibida	Ângulo da zona proibida (graus)	80

Sendo que cabe ao caminhante manter o cata-vento no sentido oposto do corredor, ao longo da trajetória em ziguezague, considerou-se uma zona proibida de valor elevado (80º) de forma a facilitar este procedimento e garantir que o controlador mantenha o mesmo corredor ao longo do percurso.

O trajeto consiste na partida da área da etapa 1 e seguir face às etapas intermédias de bolina de acordo com o feedback dado pela aplicação, acabando na área da etapa 2. A etapa 3 (de coordenadas iguais à etapa 1) serve apenas para testar a transição dos estados de bolina para o estado normal a quando da chegada à etapa 2, estando os pontos deste trajeto ilustrados na figura 6.4.

Figura 6.4: Etapas do teste do nível 2 do controlador (Google Earth, imagem de 27/6/2007) [60].

Ao longo do teste foram calculadas 4 coordenadas intermédias de bolina à medida que o controlador alternou entre os estados 2 e 3 da RdP, estando estas indicadas na tabela 6.6 pela respetiva ordem de cálculo.

Tabela 6.6: Coordenadas intermédias de bolina calculadas no teste do nível 2.

Ordem	Estado RdP	Longitude	Latitude
1	3	-9.1897955	38.616493
2	2	-9.189308	38.61681
3	3	-9.189718	38.616935
4	2	-9.189465	38.616707

Feito o teste, procedeu-se à análise dos valores registados pela aplicação de monitorização, estando a caracterização do corredor obtido pelo controlador presente na tabela 6.7 e alguns dos restantes valores presentes na tabela 6.8.

Tabela 6.7: Características do corredor calculado do teste do nível 2.

Retas		
Nome	**Declive**	**Ordenada na origem**
R_0	-1.5336788	24.522915
R_1	-1.5336788	24.523014
R_2	-1.5336788	24.522816
R_3	0.652027	44.608486
R_4	4.5	79.96944
R_5	-0.22222222	36.57455
Balizas		
Nome	**longitude**	**latitude**
B_1	-9.189499	38.616688
B_2	-9.189528	38.61667

As linhas a cinzento da tabela 6.8 correspondem a uma viragem a estibordo e as linhas brancas vão corresponder a uma viragem a bombordo face às coordenadas intermédias da tabela 6.6, dispostas na mesma ordem. A linha final marca o retorno ao estado 1 da RdP (estado *Normal*).

Tabela 6.8: Alguns valores registados pela aplicação de monitorização do teste do nível 2.

Correction angle	Rudder servo	Sail servo	Roll	Bearing	WV sym.	GPS lng	GPS lat	IOPT state	Next WP
-12.081757	1250	1287	2	156	118	-9.189697	38.61696	3	1
7.184677	1597	1003	3	176	139	-9.189713	38.616913	3	1
-61.200165	1099	1003	10	199	169	-9.189728	38.616882	2	1
-10.933502	1272	1261	3	250	121	-9.189667	38.616867	2	1
-8.972748	1360	1316	6	253	114	-9.189606	38.61685	2	1
88.0372	1996	1302	6	256	116	-9.18956	38.616867	3	1
-14.5841675	1249	1003	2	154	152	-9.189575	38.616806	3	1

Correction angle	Rudder servo	Sail servo	Roll	Bearing	WV sym.	GPS lng	GPS lat	IOPT state	Next WP
-24.748047	1250	1003	5	152	149	-9.189606	38.616776	3	1
-85.47119	1003	1003	7	172	161	-9.189636	38.616745	2	1
-0.6626892	1490	1337	6	267	111	-9.1896515	38.616714	2	1
-6.905945	1407	1450	5	260	95	-9.189606	38.616714	2	1
-6.559662	1413	1383	9	258	104	-9.189545	38.616714	2	1
-164.84344	1003	1251	3	238	122	-9.189499	38.6167	1	2

De forma semelhante ao que foi feito no teste dos níveis 1 e 3, procedeu-se à análise da trajetória no Google Earth, obtendo-se os resultados da figura 6.5.

Figura 6.5: Trajeto do teste do nível 2 com áreas aproximadas de etapa *a azul* e pontos de mudança de estado RdP *a amarelo* (Google Earth, imagem de 27/6/2007) [60].

Verificando a trajetória pode-se ver uma pequena correção abrupta no final da primeira viragem a bombordo que, como já mencionado, pode ser proveniente quer das interferências magnéticas entre sensores, quer das imprecisões do próprio GPS.

Outro ponto a ter em conta é que no caso de as margens do corredor serem iguais ao raio de etapa (6m), as retas de estreitamento de 45º não serão necessárias à navegação pois grande parte da área cortada pelo estreitamento já se encontra dentro da área de etapa. Estas só serão úteis em casos em que as margens do corredor excedam largamente o raio de etapa ou quando se definem ângulos de estreitamento de valor elevado.

Não obstante, o controlador foi capaz de tomar uma trajetória à bolina em ziguezague dentro das retas de limite do corredor, sendo capaz de no fim voltar corretamente ao estado normal de navegação face à etapa 3 (igual à etapa 1). Além disso, os valores referentes aos servos da vela e do leme estão corretos em função dos valores esperados.

Este teste foi feito por utilização do código referente ao sistema completo que ocupa 14% da memória flash, 59% da memória RAM e tem um tempo de ciclo aproximado entre os 542 e os 547 ms.

6.3. Teste completo

O terceiro e último teste de campo teve como objetivo testar um cenário misto a fim de verificar o funcionamento de todos os níveis do controlador.

Este teste é definido por três etapas num trajeto de ida e volta composto por dois troços: um primeiro troço ao longo do qual se testa mais uma vez a capacidade de navegação à bolina e um segundo troço no qual se testa a capacidade de orientação face à próxima etapa em condições de vento favorável, estando as coordenadas destas etapas definidas na tabela 6.9.

Tabela 6.9: Etapas do trajeto do teste completo

Etapa	Longitude	Latitude
1	-9.189600	38.610462
2	-9.190039	38.610476
3	-9.189600	38.610462

A fim de maximizar as chances de causar uma viragem de estado à bolina por entrada numa das zonas de estreitamento do corredor (zonas 5 e 6) fez-se alguns ajustes aos valores de calibração, conforme indicado na tabela 6.10.

Tabela 6.10: Valores de calibração do teste completo.

Valores de calibração			
Aplicação		Constante de amostragem de registo (ms)	2500
Nível 3		Contador de chegada à etapa	3
		Raio circular de etapa (m)	3
Nível 2	*Corredor*	Distância *D1* (m)	8
		Distância *D2* (m)	8
		Distância *D3* (m)	3
		Distância *D4* (m)	3
		Ângulo de estreitamento θ_1 (graus)	75
		Ângulo de estreitamento θ_2 (graus)	75
	Bolina	Ângulo de coordenada intermédia (graus)	50
	Zona Proibida	Ângulo da zona proibida (graus)	80

Ao longo do primeiro troço, no sentido 1→2, manteve-se o cata-vento fixo no sentido oposto do mesmo a fim de seguir as coordenadas intermédias resultantes do cenário à bolina. Uma vez atingida a etapa 2 voltou-se à etapa de origem (etapa 3 coincidente com etapa 1 da figura 6.6) mantendo-se o cata-vento no sentido desta a fim de emular um cenário de vento favorável.

Figura 6.6: Etapas do teste completo (Google Earth, imagem de 27/6/2007) [60].

No decurso deste teste registaram-se 4 etapas intermédias de bolina cujas coordenadas estão apresentadas na tabela 6.11.

Tabela 6.11: Coordenadas intermédias de bolina calculadas no teste completo.

Ordem	Estado RdP	Longitude	Latitude
1	2	-9.1900425	38.609985
2	3	-9.190035	38.610962
3	2	-9.190041	38.61023
4	3	-9.190039	38.610474

As características do corredor obtido encontram-se presentes na tabela 6.12.

Tabela 6.12: Características do corredor calculado do teste completo.

Retas		
Nome	**Declive**	**Ordenada na origem**
R_0	-0.008810572	38.529507
R_1	-0.008810572	38.529434
R_2	-0.008810572	38.52958
R_3	113.50001	1081.6799
R_4	-0.2962963	35.887478
R_5	0.2962963	41.333477
Balizas		
Nome	**longitude**	**latitude**
B_1	-9.190039	38.61045
B_2	-9.190039	38.610504

Os restantes resultados obtidos pela aplicação estão presentes na tabela 6.13. As linhas a cinzento representam uma viragem a bombordo (estado 2 da RdP) e as linhas a branco representam uma viragem a estibordo (estado 3), isto face às coordenadas intermédias da tabela 6.11 (na mesma ordem). Os valores referentes ao estado 1 da RdP encontram-se indicados a verde.

Tabela 6.13: Alguns valores registados pela aplicação de monitorização do teste completo.

Correction angle	Rudder servo	Sail servo	Roll	Bearing	WV sym.	GPS lng	GPS lat	IOPT state	Next WP
10.504799	1701	1997	9	100	20	-9.189606	38.610474	1	1
10.70517	1713	1251	10	149	122	-9.189621	38.61046	2	1
8.464478	1625	1251	7	147	122	-9.1896515	38.610428	2	1
113.27597	1997	1278	11	143	119	-9.189713	38.610397	3	1
89.09407	1996	1003	6	117	149	-9.189728	38.610382	3	1
8.665916	1630	1218	12	36	125	-9.189743	38.610397	3	1
0.6222954	1508	1096	12	28	132	-9.189758	38.610428	3	1
-3.4265747	1456	1003	11	24	135	-9.189774	38.61046	3	1
4.8462563	1561	1003	13	33	145	-9.189774	38.610474	3	1
9.74197	1666	1003	14	38	150	-9.189789	38.610504	3	1

Correction angle	Rudder servo	Sail servo	Roll	Bearing	WV sym.	GPS lng	GPS lat	IOPT state	Next WP
-112.61148	1003	1069	11	34	133	-9.189819	38.610565	2	1
11.604446	1750	1176	14	162	128	-9.18985	38.610565	2	1
8.824097	1635	1160	13	161	129	-9.189896	38.610504	2	1
-1.8851624	1475	1241	13	153	123	-9.189926	38.610474	2	1
-3.5713043	1454	1218	14	153	125	-9.189941	38.61046	2	1
-11.181107	1255	1230	14	151	124	-9.189972	38.610443	2	1
105.63354	1997	1038	10	154	134	-9.189987	38.610428	3	1
-3.367485	1456	1241	12	35	123	-9.190002	38.610428	3	1
-26.166344	1250	1096	13	41	132	-9.190002	38.61046	3	1
-29	1250	1003	12	61	148	-9.190018	38.610474	3	1
162.5292	1997	1003	16	69	148	-9.190033	38.61049	1	2
-42.470795	1200	1997	10	224	38	-9.190033	38.61049	1	2
1.5693665	1520	1997	12	270	32	-9.190018	38.610474	1	2
-13.307007	1250	1997	13	255	31	-9.189987	38.610474	1	2
8.106506	1616	1997	13	274	27	-9.189972	38.61049	1	2
-16.527863	1249	1997	12	249	27	-9.189941	38.61049	1	2
-6.0909424	1420	1997	13	259	31	-9.189911	38.61049	1	2
-3.5596619	1454	1997	13	261	27	-9.18988	38.61049	1	2
-5.5457153	1429	1997	12	257	26	-9.189804	38.61049	1	2
-9.227722	1352	1997	13	257	23	-9.189774	38.610474	1	2
-0.8653259	1488	1997	14	265	25	-9.189758	38.610474	1	2
-7.4260864	1397	1997	7	258	24	-9.189743	38.610474	1	2
-2.1932678	1471	1997	4	262	28	-9.189713	38.610474	1	2
42.842773	1801	1997	7	261	18	-9.189621	38.61049	1	2
58.905243	1881	1997	8	251	16	-9.189606	38.61049	1	2
16.65384	1749	1997	2	177	1	-9.18959	38.61049	1	2
-25.194427	1250	1997	0	115	4	-9.18959	38.610474	1	N/A

Registados os valores procedeu-se à análise da trajetória de forma semelhante aos testes anteriores. As mudanças de estado da RdP assim como partidas/chegadas a cada etapa estão representadas pelos pontos amarelos, conforme mostrado na figura 6.7.

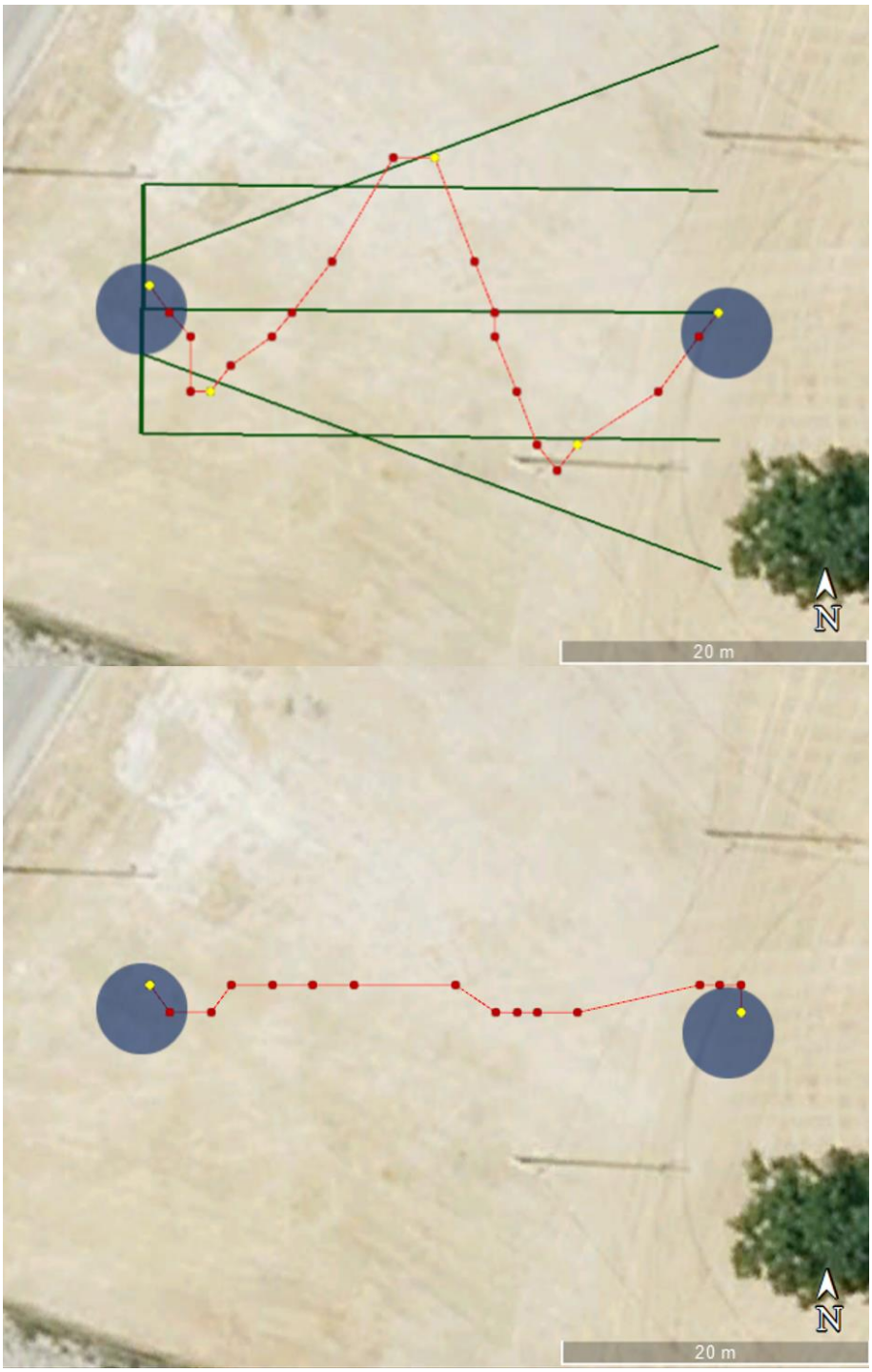


Figura 6.7: Trajeto do teste completo com troço à bolina *em cima* e troço de vento favorável *em baixo* (Google Earth, imagem de 27/6/2007) [60].

Analisando os resultados pode-se verificar que o controlador concretizou corretamente as mudanças de estado referentes à navegação à bolina durante o primeiro troço do trajeto, tanto quando o sistema se encontrou fora das retas de limite, como a quando da entrada na zona 5 excluída pela reta 4.

Na primeira viragem a bombordo, assim como antes e depois da segunda viragem para o mesmo bordo, ocorreram falhas pontuais de comunicação que resultaram num espaçamento entre pontos maior que o normal devido à filtragem de tramas incorretas por parte da aplicação. Além disto ocorreu também uma viragem abrupta a quando da aproximação para com a etapa 2.

Quanto ao último troço do trajeto pôde-se também verificar duas falhas pontuais na comunicação, culminando numa correção abrupta a quando da chegada à última etapa, sendo que esta foi influenciada pela aquisição correta do sinal quando o sistema já estava muito próximo da última área. Contudo será de notar que o raio de etapa utilizado foi muito inferior àqueles utilizados nos outros testes (3 metros), estando assim mais próximo do grau de precisão do GPS.

Apesar das imperfeições registadas, o sistema foi capaz de orientar um caminhante ao longo das etapas definidas, tanto em cenário de emulação de navegação à bolina como em cenário de emulação de vento favorável, estando os valores registados de acordo com o comportamento esperado, provando-se assim o funcionamento conjunto do controlador e da aplicação de monitorização e registo de resultados.

Capítulo 7

Conclusão e trabalho futuro

Neste capítulo faz-se a apresentação de conclusões referentes ao trabalho desenvolvido, assim como propostas úteis a projetos futuros.

O controlador desenvolvido foi capaz de cumprir todos os objetivos propostos, tendo-se assim obtido um sistema capaz de se encaminhar ao longo de um trajeto definido por várias etapas, tanto em condições de vento favorável como desfavorável.

À semelhança do que já foi testado nas dissertações eVentos 3 e eVentos 4, o sistema de controlo foi capaz de se orientar face a cada etapa individual do trajeto, ajustando os valores dos servos de acordo com a aquisição sensorial (nível 1 da estratégia de controlo), assim como de detetar a condição de bolina e realizar o cálculo de coordenadas intermédias (nível 2). Além disto, de acordo com a análise de resultados e trajetórias desempenhadas durante os testes de campo, verificou-se que o sistema desenvolvido foi também capaz de proceder a mudanças corretas de estado da RdP mediante a zona de localização no corredor à bolina, de detetar corretamente a chegada a cada etapa do trajeto (nível 3) e de ser capaz de transmitir feedback em tempo real ao utilizador por intermédio da aplicação de monitorização, pontos estes que ainda não tinham sido verificados pela elaboração de testes no exterior.

Outro ponto positivo desta dissertação são as vantagens que traz para projetos futuros relacionados com a linha eVentos. A adição do efeito do "roll" como variável de entrada do controlador difuso irá permitir maior estabilidade ao veleiro em cenários futuros de navegação e as alterações feitas ao nível 2 da estratégia de controlo permitem maior versatilidade e robustez na definição de corredores de navegação à bolina.

A modularização e comentário detalhado do código de controlo a par com a documentação produzida, quer neste documento como no guia laboratorial, permitem também uma compreensão facilitada das características do sistema.

Atendendo ao estado atual do sistema de controlo, propõem-se assim 3 melhorias possíveis para projetos que deem continuidade à dissertação corrente:

1. **Obtenção de um modelo de veleiro mais robusto**

O principal fator limitador do sistema é a dimensão reduzida dos veleiros disponíveis. Apesar de não terem sido realizados testes de navegação, verificou-se alguma dificuldade em compactar o sistema dentro das caixas de ligações, o que torna a adição de novos dispositivos de hardware uma tarefa complicada, além de poder impossibilitar a devida separação dos componentes a fim de evitar interferências magnéticas mútuas.

2. **Estratégia de evasão de obstáculos**

A fim de o veleiro melhorar a sua autonomia, recomenda-se a inclusão de estratégias que lhe permitam evitar possíveis obstáculos existentes ao longo de um percurso convencional de navegação.

3. **Definição de estratégias orientadas à competição**

Uma vez garantidas as propostas anteriores, é possível preparar o veleiro para cenários de competição, realizando uma análise da performance deste em testes de navegação a fim de proceder aos ajustes necessários à melhoria de resultados, tendo em conta o contexto específico das provas.

Bibliografia

[1] F. Le Bars and L. Jaulin, Eds., *Robotic Sailing 2013: Proceedings of the 6th International Robotic Sailing Conference*. Springer, 2014.

[2] F. M. G. Martins, "eVentos 2 - Autonomous sailboat control," Almada: Faculdade de Ciências e Tecnologia- Universidade Nova de Lisboa, 2013.

[3] T. J. de A. Pereira, "eVentos 3 – Desenvolvimento de controlador difuso para navegação autónoma de veleiro," Almada: Faculdade de Ciências e Tecnologia- Universidade Nova de Lisboa, 2015.

[4] M. F. P. Santos, "eVentos 4 - Controlador para navegação autónoma de veleiro em modo de regata," Almada: Faculdade de Ciências e Tecnologia- Universidade Nova de Lisboa, 2015.

[5] Thunder Tiger, "Naulantia 1M Racing Yacht." [Online]. Available: http://www.thundertiger.com/products-detail.php?id=11. [Accessed: 03-Aug-2020].

[6] Marina de Cascais, "Informações Náuticas: Glossário." [Online]. Available: https://www.mymarinacascais.com/info_nauticas/glossario/. [Accessed: 03-Aug-2020].

[7] Internet Archive: ANC-Associação Nacional de Cruzeiros, "Navegação à Vela." [Online]. Available: https://web.archive.org/web/20110811073433/http://www.ancruzeiros.pt/ancnav-vela.html. [Accessed: 03-Aug-2020].

[8] T. Cunliffe, *The complete yachtmaster : sailing, seamanship and navigation for the modern yacht skipper*. Adlard Coles Nautical, 2010.

[9] L. A. Zadeh and R. R. Yager, *An Introduction to Fuzzy Logic Applications in Intelligent Systems*. Springer US, 1992.

[10] L. A. Zadeh, "Fuzzy Sets," Berkeley, California, 1965.

[11] W. Reisig, *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies*. Heidelberg, Berlin: Springer-Verlag, 2013.

[12] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[13] J. Esteves, L. Gomes, and A. Costa, "Collision Avoidance System for an Autonomous Sailboat," in *IECON'2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017.

[14] J. M. F. Esteves, "Desenvolvimento de um Sistema anticolisão para um veleiro com navegação autónoma," Almada: Faculdade de Ciências e Tecnologia- Universidade Nova de Lisboa, 2017.

[15] "Pixy CMUcam5 Sensor." [Online]. Available: https://media.digikey.com/pdf/Data Sheets/Seeed Technology/101990056_Web.pdf. [Accessed: 22-Nov-2020].

[16] J. C. C. P. F. Andrade, "Prova de evasão de obstáculos em competição de veleiros com navegação autónoma," Almada: Faculdade de Ciências e Tecnologia- Universidade Nova de Lisboa, 2019.

[17] R. A. C. Santos, "Prova de varrimento de área em competição de veleiros com navegação autónoma," Almada: Faculdade de Ciências e Tecnologia- Universidade Nova de Lisboa, 2019.

[18] "World Robotic Sailing Championship & International Robotic Sailing Conference, Sep 4-8, Horten, Norway." [Online]. Available: https://www.wrsc2017.com/. [Accessed: 31-Jan-2020].

[19] Arduino, "Arduino MEGA 2560 & Genuino MEGA 2560." [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardMega2560. [Accessed: 10-Aug-2020].

[20] Argent Data Systems, "Weather sensor assembly." [Online]. Available: https://www.sparkfun.com/datasheets/Sensors/Weather/Weather Sensor Assembly..pdf. [Accessed: 10-Aug-2020].

[21] Robot Electronics, "CMPS10 - Tilt Compensated Compass Module." [Online]. Available: http://www.robot-electronics.co.uk/htm/cmps10doc.htm. [Accessed: 10-Aug-2020].

[22] "Receptor GPS de 20 canais EM-406A_M." [Online]. Available: https://s3-sa-east-1.amazonaws.com/multilogica-files/imagens/SparkFun/Receptor_GPS_de_20_canais_EM-406A_M.jpg. [Accessed: 10-Aug-2020].

[23] "GPS tracker, wearable devices Manufacturer – GlobalSat WorldCom Corp." [Online]. Available: https://www.globalsat.com.tw/. [Accessed: 10-Aug-2020].

[24] National Marine Electronics Association, "NMEA." [Online]. Available: https://www.nmea.org. [Accessed: 12-Aug-2020].

[25] Instituto de Microelectrónica de Sevilla, “Fuzzy Logic Design Tool.” [Online]. Available: http://www2.imse-cnm.csic.es/Xfuzzy/Xfuzzy_3.3/index.html. [Accessed: 12-Aug-2020].

[26] austriamicrosystems, “AS5161, Rotary Sensor 12-bit rotary position sensor with PWM output and overvoltage protection.” [Online]. Available: http://ams.com/eng/Products/Magnetic-Position-Sensors/Angle-Position-On-Axis/AS5161. [Accessed: 18-Aug-2020].

[27] “MEDIATEK - 3329 Datasheet.” [Online]. Available: https://drive.google.com/file/d/0B_dHj7E2weiiNmUzNDA3OTktNTNhNy00Y2Y5LTg0YTQtMzIyNzJhZmFiNjcy/view?hl=en. [Accessed: 18-Aug-2020].

[28] Speedstudio, “Wi-Fi Shield.” [Online]. Available: https://www.seeedstudio.com/item_detail.html?p_id=1220. [Accessed: 18-Aug-2020].

[29] L. Gomes, M. Santos, T. Pereira, A. Costa, F. Moutinho, and R. Mota, “Model-Based Development of an Autonomous Sailing Yacht Controller,” *2015 IEEE Int. Conf. Auton. Robot Syst. Compet.*, pp. 103–108, 2015.

[30] GRES: R&D Group on Reconfigurable and Embedded Systems - Universidade Nova de Lisboa, “IOPT Tools.” [Online]. Available: http://gres.uninova.pt/IOPT-Tools/login.php. [Accessed: 21-Aug-2020].

[31] Google, “Google Maps.” [Online]. Available: https://www.google.pt/maps/. [Accessed: 18-Aug-2020].

[32] desmos, “Desmos Graphing Calculator.” [Online]. Available: https://www.desmos.com/. [Accessed: 23-Aug-2020].

[33] “Clube Náutico de Almada.” [Online]. Available: http://www.cnalmada.com/. [Accessed: 23-Aug-2020].

[34] “CINAV, Centro de investigação naval.” [Online]. Available: http://escolanaval.marinha.pt/pt/investigacao. [Accessed: 23-Aug-2020].

[35] Instituto Newton C. Braga, “Como funcionam os sensores ultrassónicos.” [Online]. Available: http://www.newtoncbraga.com.br/index.php/como-funciona/5273-art691. [Accessed: 23-Aug-2020].

[36] Atmel, “ATmega2560.” [Online]. Available: http://www.atmel.com/devices/atmega2560.aspx. [Accessed: 10-Aug-2020].

[37] “arduinomega2560.” [Online]. Available: https://hifiduino.files.wordpress.com/2012/03/arduinomega2560_r2_front.jpg. [Accessed: 10-Aug-2020].

[38] Arduino, “Arduino IDE.” [Online]. Available: https://www.arduino.cc/en/main/software. [Accessed: 10-Aug-2020].

[39] “MT3339 All-in-One GPS Datasheet.” [Online]. Available: https://s3-ap-southeast-1.amazonaws.com/mediatek-labs-imgs/downloads/51c3cf5aaee074767384ab70d4f7d602.pdf. [Accessed: 29-Aug-2020].

[40] “Adafruit Ultimate GPS.” [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps.pdf. [Accessed: 29-Aug-2020].

[41] “Adafruit_GPS - An interrupt-based GPS library for no-parsing-required use.” [Online]. Available: https://github.com/adafruit/Adafruit_GPS. [Accessed: 29-Aug-2020].

[42] “GPS - NMEA sentence information.” [Online]. Available: http://aprs.gids.nl/nmea/. [Accessed: 29-Aug-2020].

[43] National Geospatial-Intelligence Agency, “WORLD GEODETIC SYSTEM 1984 (WGS 84).” [Online]. Available: https://earth-info.nga.mil/GandG/update/index.php?dir=wgs84&action=wgs84#tab_wgs84-res. [Accessed: 29-Aug-2020].

[44] ServoDatabase, “Hitec HS-785HB - 3.5 Turn Sail Winch Servo.” [Online]. Available: https://servodatabase.com/servo/hitec/hs-785hb. [Accessed: 29-Aug-2020].

[45] ServoDatabase, “Zebra ZS-S2113 Servo.” [Online]. Available: https://servodatabase.com/servo/zebra/zs-s2113. [Accessed: 29-Aug-2020].

[46] DFRobot, “APC220 Radio Communication Module.” [Online]. Available: https://www.dfrobot.com/product-57.html. [Accessed: 29-Aug-2020].

[47] Amazon, “Tolako 1000M APC220 Wireless RF Module: Radio Communication Module for Arduino.” [Online]. Available: https://www.amazon.com/Tolako-APC220-Wireless-Communication-Arduino/dp/B01ELMLYCA/. [Accessed: 29-Aug-2020].

[48] ITEAD, “ITEAD Joystick shield.” [Online]. Available: https://www.itead.cc/wiki/ITEAD_Joystick_shield. [Accessed: 30-Aug-2020].

[49] Apache Software Foundation, “Apache NetBeans IDE.” [Online]. Available: https://netbeans.org/. [Accessed: 31-Aug-2020].

[50] “MinGW - Minimalist GNU for Windows.” [Online]. Available: https://sourceforge.net/projects/mingw/. [Accessed: 01-Sep-2020].

[51] “R&D Group on Reconfigurable and Embedded Systems.” [Online]. Available: http://gres.uninova.pt/. [Accessed: 21-Aug-2020].

[52] F. Pereira and L. Gomes, “IOPT User Manual Version 1.1,” 2014. [Online]. Available: http://gres.uninova.pt/iopt_usermanual.pdf. [Accessed: 26-Nov-2020].

[53] L. Gomes, A. Costa, D. Fernandes, H. Marques, and F. Anjos, “Improving instrumentation support and control strategies for autonomous sailboats in a regatta contest,” in *ISRC’16 - 9th International Robotic Sailing Conference*, 2016.

[54] R. Larson, *Precalculus: A Concise Course*, 2nd Ed. Cengage Learning, 2010.

[55] E. W. Weisstein, “‘Rotation Matrix.’ From MathWorld--A Wolfram Web Resource.” [Online]. Available: http://mathworld.wolfram.com/RotationMatrix.html. [Accessed: 07-Oct-2020].

[56] Fazecast Inc., “jSerialComm.” [Online]. Available: https://fazecast.github.io/jSerialComm/. [Accessed: 16-Sep-2020].

[57] “IEEE-754 Floating Point Converter.” [Online]. Available: https://www.h-schmidt.net/FloatConverter/IEEE754.html. [Accessed: 17-Sep-2020].

[58] F. Anjos, “Guia eVentos,” 2020. [Online]. Available: http://gres.uninova.pt/~lugo/veleiros/NavegacaoAutonomaVeleiros_ficheiros/Frota_Guia.pdf. [Accessed: 22-Nov-2020].

[59] L. Gomes, A. Costa, F. Moutinho, and R. Mota, “Attracting students to Engineering through Autonomous Sailing Yacht development,” in *ICIT 2015 - 2015 IEEE International Conference on Industrial Technology*, 2015.

[60] Google, “Google Earth.” [Online]. Available: https://www.google.com/intl/pt-PT/earth/. [Accessed: 08-Oct-2020].

Anexo

A1. Esquema de ligações do sistema

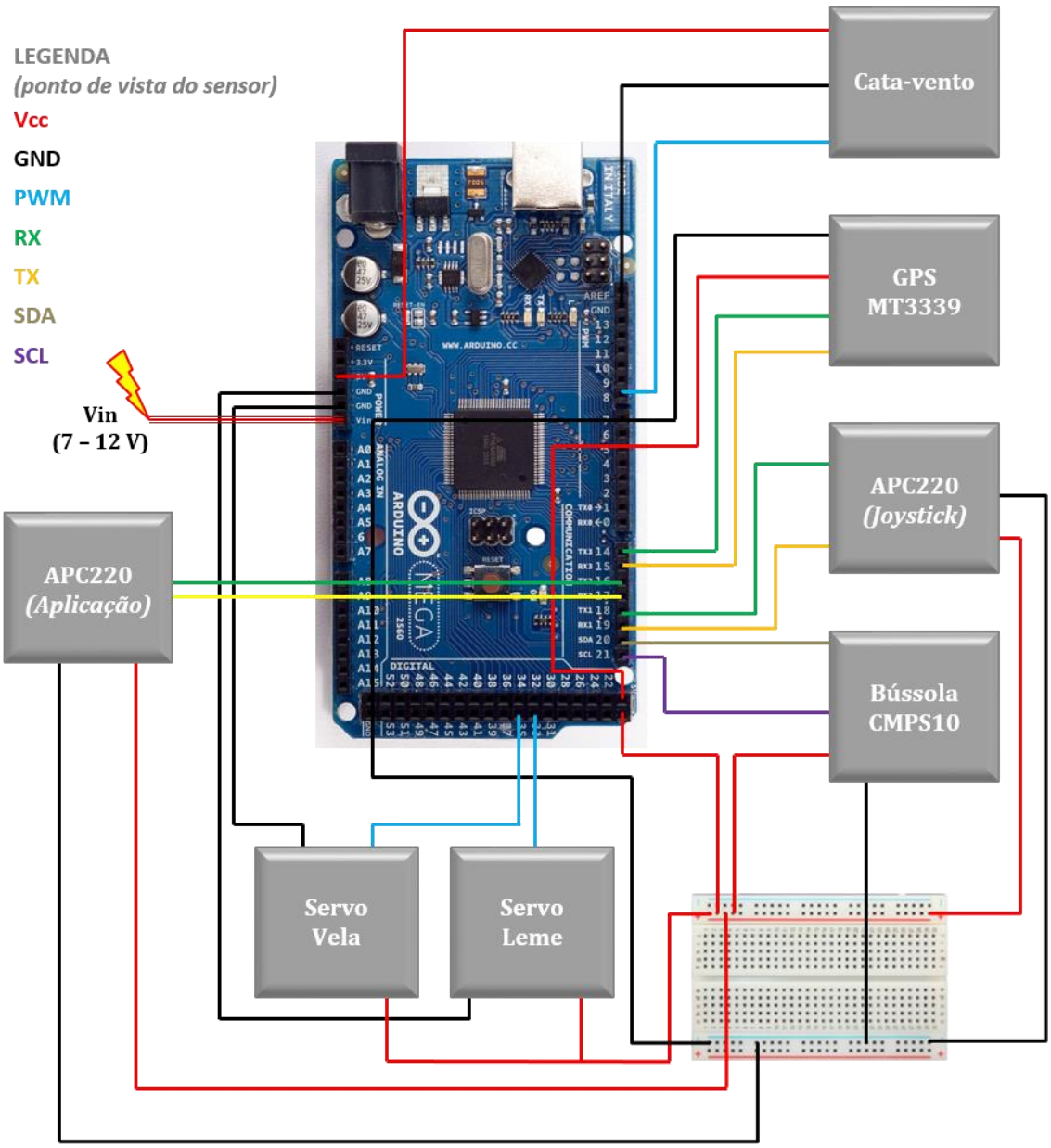


Figura A1.1: Esquema de ligações do sistema [37].

A2. Adaptação do código da RdP

No interface da IOPT Tools, após se ter a rede selecionada e clicado na funcionalidade "C code", vai-se obter uma conjunto de ficheiros de código C que se terão que adaptar ao controlador já existente, como a seguir descrito:

1. Adição dos ficheiros "net_io.c", "net_types.h", "net_functions.c" e "net_exec_step.c" à pasta do projeto. Alterar as extensões dos ficheiros source para ".cpp". Guardar também o ficheiro "net_main.c" à parte;
2. No ficheiro "net_io.cpp" declarar dois apontadores do tipo *Joystick_rcv* e *Level2*, como na figura A2.1 (classes detentoras das variáveis dos sinais de entrada);

```
#include <stdlib.h>
#include "net_types.h"

#ifdef ARDUINO
#include <Arduino.h>
#define ANALOG_IN_MAX  1023
#define ANALOG_OUT_MAX 255
#else
#define INPUT    0
#define OUTPUT   1
#define ANALOG_IN_MAX  1023
#define ANALOG_OUT_MAX 1023
extern void pinMode( int, int );
extern int  digitalRead( int );
extern void digitalWrite( int, int );
extern int  analogRead( int );
extern void analogWrite( int, int );
#endif

Joystick_rcv* joystick_pt;
Level2* level2_pt;

// Remote IcE/Debug forced values:
#ifdef HTTP_SERVER
iopt_param_info *input_fv = NULL, *output_fv = NULL;
#endif
```

Figura A2.1: Declaração dos apontadores relevantes em "net_io".

3. No mesmo ficheiro, declarar como parâmetros dois apontadores do mesmo tipo na função *eVentos5_IOPT_InitializeIO* e igualar estes àqueles anteriormente declarados no corpo da função. De seguida, na função *eVentos5_IOPT_GetInputSignals*, associar os argumentos das respetivas classes aos sinais de entrada por intermédio dos apontadores (figura A2.2);

```
/* Executed just once, before net execution starts: */
void eVentos5_IOPT_InitializeIO(Joystick_rcv* joystick_para, Level2* level2_para)
{
  joystick_pt = joystick_para;
  level2_pt = level2_para;
}


/* Read all hardware input signals and fill data-structure */
void eVentos5_IOPT_GetInputSignals(
  eVentos5_IOPT_InputSignals* inputs,
  eVentos5_IOPT_InputSignalEvents* events )
{
  inputs->InS_Wind = level2_pt->into_Wind;
  inputs->InS_ManualCtrl = joystick_pt->cnt_status;
  inputs->InS_Zone = level2_pt->zone;
#ifdef HTTP_SERVER
  if ( input_fv != NULL ) force_eVentos5_IOPT_Inputs( input_fv, inputs );
#endif
}
```

Figura A2.2: Adaptação das funções no ficheiro "net_io.cpp".

4. Incluir no ficheiro "net_types.h" os headers das classes do joystick e do nível 2, conforme presente na figura A2.3;

```
/* Net eVentos5_IOPT - IOPT */
/* Automatic code generated by IOPT2C XSLT transformation. */


#ifndef __eVentos5_IOPT_DEFS
#define __eVentos5_IOPT_DEFS

#include "Level2.h"
#include "Joystick_RCV.h"

#define TRACE_CONT_RUN       (-1)
#define TRACE_PAUSE          0
#define TRACE_SINGLE_STEP    1
#define TRACE_N_STEPS(n)     (n)

#define MODEL_NAME    eVentos5_IOPT
#define MODEL_NAME_STR    "eVentos5_IOPT"
#define MODEL_VERSION   "2020-09-30 11:26:42"
#define MODEL_N_INPUTS    3
#define MODEL_N_OUTPUTS   1
#define MODEL_N_PLACES    4
#define MODEL_N_TRANSITIONS 10

#ifdef __cplusplus
extern "C" {
#endif
```

Figura A2.3: Adição de headers no ficheiro "net_types.h".

5. No mesmo ficheiro, igualar a declaração da função denominada *eVentos5_IOPT_InitializeIO* de acordo com o que se fez em "net_io.cpp" (figura A2.4);

```
extern void eVentos5_IOPT_InitializeIO(Joystick_rcv* joystick_para, Level2* level2_para);
extern void eVentos5_IOPT_GetInputSignals( eVentos5_IOPT_InputSignals* inputs, eVentos5_IOPT_Inpu
extern void eVentos5_IOPT_PutOutputSignals( eVentos5_IOPT_PlaceOutputSignals* place_out, eVentos5
extern void eVentos5_IOPT_LoopDelay();
extern int eVentos5_IOPT_FinishExecution( eVentos5_IOPT_NetMarking* marking );

extern eVentos5_IOPT_NetMarking* get_eVentos5_IOPT_NetMarking();
extern eVentos5_IOPT_InputSignals* get_eVentos5_IOPT_InputSignals();
extern eVentos5_IOPT_PlaceOutputSignals* get_eVentos5_IOPT_PlaceOutputSignals();
extern eVentos5_IOPT_EventOutputSignals* get_eVentos5_IOPT_EventOutputSignals();
extern eVentos5_IOPT_TransitionFiring* get_eVentos5_IOPT_TransitionFiring();

#ifdef __cplusplus
};
#endif


#endif
```

Figura A2.4: Declaração da função *eVentos5_IOPT_InitializeIO* em "net_types.h".

6. No ficheiro principal do projeto Arduino, incluir o header "net_types.h" e copiar as declarações iniciais do ficheiro "net_main.c", ignorando aquelas relacionadas com o servidor, de acordo coma figura A2.5;

```
#include <Arduino.h>
#include <Adafruit_GPS.h>
#include "Consts.h"
//#include "Joystick_RCV.h" //incluidos em net_types.h
//#include "Level2.h"
#include "WV.h"
#include "CMP.h"
#include "GPS_mt3339.h"
#include "Level3.h"
#include "Pre_proc.h"
#include "Level1.h"
#include "Servos.h"
#include "COM_emitter.h"
#include "net_types.h"
```

```
//IOPT
int trace_control = TRACE_CONT_RUN;
static eVentos5_IOPT_NetMarking marking;
static eVentos5_IOPT_InputSignals inputs, prev_inputs;
static eVentos5_IOPT_PlaceOutputSignals place_out;
static eVentos5_IOPT_EventOutputSignals ev_out;
```

Figura A2.5: Adição de header "net_types.h" e declarações do "main file" do projeto Arduino, copiados de "net_main.c".

7. Colocar as ações das funções setup e loop presentes no ficheiro gerado "net_main.c" dentro de duas novas funções no ficheiro principal do projeto Arduino. Estas funções deverão ser chamadas tanto no setup como no loop do ficheiro principal do projeto. Isto faz-se a fim de as funções de origem não se confundirem com as funções do Arduino de mesmo nome. Na função setup copiada, colocar como parâmetros da função *eVentos5_IOPT_InitializeIO* referências relativas às instâncias do nível 2 e do joystick (neste caso "&js" e "&l2"), como indicado na figura A2.6;

```
/*
   Funcoes IOPT
*/
//iniciacao
void iopt_setup() {
  createInitial_eVentos5_IOPT_NetMarking( &marking );
  init_eVentos5_IOPT_OutputSignals( &place_out, &ev_out );
  eVentos5_IOPT_InitializeIO(&js, &l2);
  eVentos5_IOPT_GetInputSignals( &prev_inputs, NULL );
}
//loop/iteracao
void iopt_loop() {
  if ( trace_control != TRACE_PAUSE )
    eVentos5_IOPT_ExecutionStep( &marking, &inputs, &prev_inputs, &place_out, &ev_out );
  else eVentos5_IOPT_GetInputSignals( &inputs, NULL );
  if ( trace_control > TRACE_PAUSE ) --trace_control;

  eVentos5_IOPT_LoopDelay();
}
```

Figura A2.6: Adaptação das funções *setup* e *loop*.

8. Copiar a definição das funções que se encontram no fim de "net_main.c" para o fim do ficheiro principal do Arduino (figura A2.7).

```
//definicoes extra IOPT (do main file)
eVentos5_IOPT_NetMarking* get_eVentos5_IOPT_NetMarking()
{
  return &marking;
}

eVentos5_IOPT_InputSignals* get_eVentos5_IOPT_InputSignals()
{
  return &inputs;
}

eVentos5_IOPT_PlaceOutputSignals* get_eVentos5_IOPT_PlaceOutputSignals()
{
  return &place_out;
}

eVentos5_IOPT_EventOutputSignals* get_eVentos5_IOPT_EventOutputSignals()
{
  return &ev_out;
}
```

Figura A2.7: Definições finais das funções a passar para o ficheiro principal.

A fim de se saber o lugar atual da situação do controlador na rede de Petri faz-se uso da função *get_eVentos5_IOPT_PlaceOutputSignals()->OuS_place* que retorna um valor do tipo "unsigned integer".